


University of South Wales



2059405

*Bound by*  
 **Abbey**  
**Bookbinding Co.**

116 Cathays Terrace, Cardiff CF24 4HY  
South Wales, U.K. Tel: (029) 20395882

# **An investigation into the use, application and evaluation of intelligent agents**

***Mike Reddy***

**PhD Overview Report**

***May 1999***

# **An investigation into the use, application and evaluation of intelligent agents**

***Mike Reddy***

**A submission presented in partial fulfillment of the requirements  
of the University of Glamorgan/Prifysgol Morgannwg  
for the degree of Doctor of Philosophy**

***May 1999***

To my unborn, Stealth Baby  
The Star Child of Sun and Moon.

And to my Victoria, who I call "Vic",  
For loving me through thin and thick!

# **Abstract**

This overview report comprises two projects linked by the theme of the application of intelligent agents. The first project covers the development of RAPIDO, a rapid prototyping toolkit for the development and evaluation of agent applications. The papers and technical reports included for this project look at the implementation and application of the RAPIDO toolkit to a specific test case for evaluation purposes. The second project considers the use of an agent-based simulation of the Internet, called WebAgent, to explore agent-based solutions to improving network performance. The publications present the experimental methodology of WebAgent, and the results of evaluating Expl, an adaptive agent for intelligent control of dynamic caching strategies for web servers and clients.

The knowledge gained during the course of these two projects has been published in refereed papers included within the accompanying portfolio. The production of Multi-Agent Systems (MAS), particularly Distributed Artificial Intelligence (DAI) applications, is dependent upon the underlying paradigm. The generic approach to the specification of agent applications, and the object-oriented rapid prototyping technique, have allowed different implementations of a specific problem domain to be evaluated in order to determine the best architecture. The application of agent-based simulation to the field of web document caching has both introduced a new tool for performing evaluations of such techniques and has helped towards the proposal of a new approach, based upon intelligent, adaptive agents. The future of such directions promises to offer far greater application of these ideas to the regulation and management of networks in general.

# Table of Contents

<i>Abstract</i> .....	<i>i</i>
<i>Declaration and Certificate of Research</i> .....	<i>iv</i>
<i>Acknowledgements</i> .....	<i>v</i>
<b>1.0 Introduction</b> .....	<b>1</b>
1.1 Agents and Agency: A brief history.....	1
1.2 Contents of the Portfolio.....	6
1.3 Author's role in the work .....	8
<b>2.0 RAPIDO</b> .....	<b>10</b>
2.1 Introduction .....	10
2.2 Rationale for RAPIDO .....	11
2.3 Work done on RAPIDO.....	12
2.4 Description of the RAPIDO papers.....	17
2.5 Conclusions and Future Work.....	24
<b>3.0 WebAgent</b> .....	<b>26</b>
3.1 Introduction .....	26
3.2 Rationale for WebAgent.....	30
3.3 Work done on WebAgent.....	32
3.4 Description of the WebAgent papers .....	40
3.5 Conclusions and Future Work.....	44
<b>4.0 Commentary</b> .....	<b>51</b>
4.1 Link between the two projects.....	51
4.2 Contribution to Knowledge .....	52
4.3 Future Work .....	53
<b>5.0 References</b> .....	<b>55</b>
<b>Appendix - Portfolio of Papers</b> .....	<b>61</b>

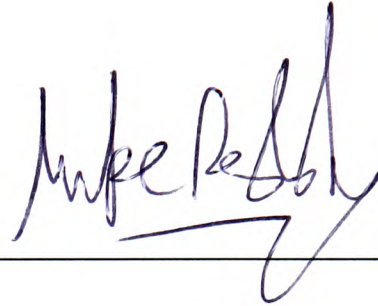
# Table of Figures

<b>Figure 2.3.1 – The architecture of RAPIDO .....</b>	<b>13</b>
<b>Figure 2.3.2 – RAPIDO top-level interface (Creating AMNESIA).....</b>	<b>15</b>
<b>Figure 2.3.3 – RAPIDO agent interface (Creating the test agent) .....</b>	<b>15</b>
<b>Figure 2.3.4 – RAPIDO skill interface (Creating and defining visible skills).....</b>	<b>16</b>
<b>Figure 2.3.5 – RAPIDO rule interface (Setting rules for memory_impaired).....</b>	<b>16</b>
<b>Figure 3.3.1 – Structure of the WebAgent Testbed.....</b>	<b>36</b>

## Declaration

This is to certify that neither this overview report, nor any part of it has been presented or is being currently submitted for any degree other than the degree of Doctor of Philosophy at the University of Glamorgan.

Candidate

A handwritten signature in blue ink, appearing to read 'Mike Kelly', written over a horizontal line.

## Certificate of Research

This is to certify that except where specific reference is made, the work presented in this overview report is the result of the investigation undertaken by the candidate.

Candidate

A handwritten signature in blue ink, appearing to read 'Mike Kelly', written over a horizontal line.

Director of Studies

\_\_\_\_\_



# Acknowledgements

I would like to take this opportunity to thank my Director of Studies, Professor Bryan Jones of Glamorgan University, and my Second Supervisor, Dr. Graham Fletcher, of Aberdeen University. Throughout the project, they have guided the work with great skill and patience. For the latter especially, I am grateful.

I must also extend thanks to Mr. Greg O'Hare of University College Dublin, my original Project Supervisor while I was studying at UMIST in Manchester for his incredible support when illness prevented the completion of my studies. Without his direction and focus the work on RAPIDO would never have been. I hope that he will share my relief that the ten years that this particular albatross has been round my neck are finally over.

I would also like to acknowledge Professor Peter Hodson and the School of Computing Staff, particularly Dr. Jim Moon, who have supported the rekindling of my research. I hope that this document will repay their faith in me over the years.

Last, but not least, I would like to recognise the love and support of my wife, Victoria Jones, whose never-ending pot of enthusiasm inspired me, even at 3am in the morning!

## **1.0 Introduction**

This is the overview report for a PhD by Portfolio in the application of intelligent agents and agent technology. The accompanying portfolio covers two projects: one in the development of a toolkit for building agent applications, the other on the use of agent-based simulations of the Internet. As the main theme of the research is that of agents and agency, these terms are defined in Section 1.1. A list of the documents that make up the portfolio is presented in Section 1.2, followed by Section 1.3, a statement of the author's contribution to the work.

### **1.1 Agents and Agency: A brief history**

As the two projects described in this report span ten years of rapid growth and development in the field of agent research, a historical perspective is useful. Since the 1970s, the field has evolved through three chronological periods: Distributed Artificial Intelligence; Multi-Agent Systems; and, more controversially, Agent Technology. These are now described to give a context to the two research projects, which will be described later.

#### **1.1.1 Distributed Artificial Intelligence**

The concept of agency is rooted in Distributed Artificial Intelligence (DAI), which branched away from main stream AI, due (in part) to inspiration from the fields of distributed, concurrent and real-time systems. This marked a distinct trend towards decentralisation, which was prevalent in Computer Science [O'Hare (1987)]. DAI proposed the use of logically distinct, possibly concurrent Knowledge-Based Systems (KBS), or Agents, which communicated with each other in a collaborative manner, in order to produce solutions to complex problems [Hern (1987)]. However, the independent production of separate components could have imposed arbitrary and artificial boundaries upon the expertise stored. Furthermore, the heterogeneous nature of some domains prevented the co-operative solution of complex problems that required knowledge from a broad spectrum of sources.

DAI researchers identified three areas of major importance: global coherence, knowledge representation and communications [Hern (1987)]. These three areas were closely inter-related, in that the coherence of a system was dependent upon the information being shared, while the representation of knowledge was shaped by the method used for transportation

between agents. Another feature of these problems was that they existed on at least two levels - the underlying philosophy at the higher level and the lower level implementation.

In fact, DAI researchers were often accused of 're-inventing the wheel' in their attempts to overcome some of these obstacles. The minimisation of communications [Davis & Smith (1983)] and completeness of messages to improve efficiency [Durfee et al (1987)] mirror the protocol of Loose Coupling and High Cohesion which were given priority in Software Engineering [Pressman (1982)].

Furthermore, early DAI paradigms paralleled the development of communications in real-time systems quite strongly: The HEARSAY speech recognition system [Reddy et al (1973); Lesser et al (1975)] was the first implementation of a distributed problem-solving paradigm, based upon the Blackboard Model [Englemore & Morgan (1988)]. A direct analogy can be drawn with the use of a common memory store for linking multiple processes [Young (1982)]. An attempt to simplify interaction between nodes was made in the BEINGS system [Lenat (1975)], by advocating a common structure divided into dedicated parts. The Actor model [Hewitt (1977)] allowed point-to-point communication and a restricted acquaintance list of potential links to prevent overloading [Martin (1981)]. Both these approaches are similar to the implementation of concurrent procedures (or processes) being structured into hierarchies of sub-processes [Young (1982)], and the Monitor concept of modular construction and enforced interfacing [Hansen (1973); Hoare (1974)].

More recently, DAI paradigms have emphasised meta-data transmission, rather than the use of global variables - ranging from plan sharing [Rosenschein (1986)], to complex message passing mechanisms [Cohen & Perrault (1979); Allen & Perrault (1980); Appelt (1985)] - This passing of messages for control and synchronisation is also true for the Rendezvous mechanism found in CSP [Hoare (1978)], DP [Hansen (1978)] and ADA [Ichbiah (1980)], which allows for synchronous and asynchronous communications based upon 'guarded commands' [Dijkstra (1975)].

Lesser and Corkill (1981) have proposed the Functionally Accurate / Co-operating model (FA/C) to enable total independence by allowing partial hypotheses and uncertain data to be passed between agents. An especially impressive feature of FA/C was the ability to recover from partial failure of planned activities. Competition was examined in the Contract Net Approach [Smith (1978)], where communication of bids and tenders enabled dynamic task decomposition. This was the first paradigm to be proposed, which embraced the concepts of competition and tendering. Finally, the ACC system (Architecture for Control and Communications) [Yang et al (1985)] allowed for self-reconfiguration of agents by providing a central 'meta' database of agent expertise. This allowed for dynamic optimisation using a question and answer mechanism to reduce processor and communication overheads.

Two broad classes of DAI systems have been identified : Task sharing and Result sharing [Smith & Davis (1981)]. Common to all these systems was the assumption that co-operation is justified because distributed problem solvers have only one function to perform, unlike other distributed systems which are multi-functional; they are therefore said to operate under the 'Benevolent Expert' metaphor. This assumption, that experts should always co-operate, has been challenged by [Genesereth et al (1984); Rosenschein & Genesereth (1984)]. They have offered the 'Competing Experts' metaphor as a counter-proposal, but the practicality of this approach has not been extensively tested. Steeb et al (1981) have provided an overview of different organisational structures for DAI where there is some global problem to solve and, therefore, some centralised control. The reader is recommended Bond & Gasser (1988) and O'Hare & Jennings (1996) for a fuller consideration of the origins of DAI.

### **1.1.2 Multi Agent Systems**

Multi-Agent Systems (MAS) were inspired by research into complex social structures, including such work as Speech Acts [Searle (1969)]. Durfee et al (1987) described the MAS architecture as "a loosely coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities." Most examples consisted of simple agents acting autonomously and, in most cases, communicating and co-operating, rather than competing. However, there was no clear definition of what an agent is: "In the AI literature there is not always a clear distinction between an agent and a process"[Pebody (1993)].

The use of the term ‘agent’ had become common within many academic publications and publicity material for commercial programs (especially Internet search applications). Clearly, there is a continuum of agency along a number of orthogonal axes: In fact, the literature has become infected with ‘adjectivitis’ in an attempt to distinguish more clearly between different uses of the term; prefixes such as ‘intelligent’, ‘cognitive’, ‘autonomous’, ‘situated’, ‘deliberative’, ‘reactive’, ‘software’, ‘mobile’, etc., are not exhaustive.

At present, research has split into a number of key areas: Agent theories and architectures (This includes interface protocols and social models of interaction); Agent languages and toolkits (This work is principally aimed at aiding the development and evaluation of MAS); and Agent Applications (This covers the practical application of agent-based techniques).

Much theoretical work at present is focussed upon models of belief, desires and intentions (BDI) to enable cognitive agents to make plans [Wooldridge & Jennings (1994)]. However, the alternative of ‘situated’ agents, which behave reactively within the context of their environment, are often accused of ‘jumping upon the agent bandwagon’. Demazeau and Müller (1991) have distinguished between deliberative and reactive agents: the former requiring a complex architecture that involves symbolic representation and planning to choose the correct behaviour; the latter can be simpler in structure, using perception to determine which (potentially pre-programmed) action to perform, without a formal model of the environment. An alternative, more generous view of reactive agents, which does not rule out some cognitive ability, was given by Fisher & Wooldridge (1993): “a reactive system is one which cannot adequately be described in terms of ‘initial’ and ‘final’ states.”

Although reactive strategies have been applied almost exclusively to simulation and robotics, they have also been proposed as a technique for problem solving [Ferber (1989); Connah (1991)]. However, reactive multi-agent systems lack some of the theoretical underpinning of deliberative agent architectures. While recent research has looked at the emergent properties of reactive systems, there are no guidelines for design and development of applications. It is commonly believed that reactive systems must form collectives to perform effectively:

“...reactive agents need companionship. They cannot work isolated and they usually achieve their tasks in groups. Reactive agents are situated: they do not take past events into account, and cannot foresee the future” [Ferber (1996)].

Reactive agents that exhibit learning capabilities, or evolve (such as artificial life systems), can clearly “take past events into account,” if only implicitly. Furthermore, the Subsumption architecture, is an example of a reactive strategy in which the agent (a mobile robot) is isolated’, but still situated in the environment it is navigating [Brooks & Connell (1986)]. Ferber was, therefore, only correct in that reactive systems do not cogitate on the past; they are not afforded the luxury of ‘contemplating their navels’, as their environments require instant responses rather than provably optimal ones. For more recent coverage of MAS research the reader is referred to Singh et al (1994). Jennings (1994) describes some real world applications of MAS techniques.

### **1.1.3 Agent Technology**

Agent technology covers the recent migration of agent-based techniques from experimental testbeds to specific ‘real world’ problems of varying complexity. This development has exhibited itself in two ways: Agent-Based Simulation Environments (ABSE), which enable models of social or co-operative behaviour to be explored within realistic virtual environments; and Intelligent Agents (IA), which attempt to implement intelligent solutions to problems that require some degree of autonomy, but often forego the co-operation or communication issues involved with MAS architectures, by interacting with the environment rather than through it.

The potential for using IAs and ABSEs for problem solving is great, but relatively unexplored. Most existing systems have been designed to study aspects of agent behaviour, or whether agents are viable for solving particular problems. However, these investigations have not looked at how they can be used as general evaluation tools in themselves to solve ‘real world’ problems. Yet, ABSEs allow robust, yet flexible, representations of complex environments to be developed from relatively simple components by implementing individual behaviour in detail and the environment. In general:

“Such simulations are based on the construction of a microworld where particular hypotheses can be explored, and experiments can be repeated and controlled in a similar way that real experiments are done in a real laboratory... This can be done by experimenting about the minimal conditions given at the microlevel which are necessary to observe phenomena at the macrolevel” [Ferber (1996)].

This approach embraces the concept of ‘weak’ agency [Moon (1997)] - originally expounded as “A weaker notion of agency”, the lesser alternative, by implication, to the notion of ‘strong’ agency advocated in [Wooldridge & Jennings (1994)] - in that it emphasises the use of agent-based techniques as a general problem solving tool. Opinion is split as to whether the strong ‘v’ weak agent debate is a fundamental issue in current research:

“... ‘Weak’ agency may not be the long term goal of the advocates of MAS, but it may solve some immediate problems. Success in this area may well support future research, rather than undermining the ideals” [Moon (1997)].

For this report, the discussion of agency is pragmatic, and leans in favour of the more pragmatic, ‘weak’ agent perspective. The author defines an agent as being an autonomous system, which performs a useful task in a non-deterministic fashion, either in a complex way, or in a complex setting. This precludes simple objects or processes from being included as agents, as their behaviour is completely pre-determined. This being so, it does not prohibit simplicity in the implementation of an agent where this reflects reality or practicality. Moon (1997) points out that we should “...use intelligent components to model the parts of the system where a discrete solution is hard to provide and use mathematical or functional models where they are more apt.”

## **1.2 Contents of the Portfolio**

The following papers and technical reports are included in the Appendices:

- [1] **Reddy, M. & O’Hare, G.M.P. (1991)**, “The blackboard model: a survey of its application” in ‘Artificial Intelligence Review,’ Vol. 5, No. 3, pp169-186.
- [2] **Reddy, M. & O’Hare, G.M.P. (1990)**, “CoCo-POP A Development Testbed for prototyping Distributed Knowledge-Based Systems,” Technical Report AI-90-3 (August 1990), Department of Computation, UMIST University, Manchester.

- [3] **Reddy, M. (1990)**, "A Proposal for an ACTOR-Based Extension to CoCo-POP – CAST," Technical Report AI-90-4 (October 1990), Department of Computation, UMIST University, Manchester.
- [4] **Reddy, M. (1990)**, "CAST – Progress for the Implementation of Actor Communications," Technical Report AI-91-1 (January 1991), Department of Computation, UMIST University, Manchester.
- [5] **Reddy, M. (1990)**, "CoCo-POP Generic System Capture," Technical Report AI-91-2 (March 1991), Department of Computation, UMIST University, Manchester.
- [6] **Reddy, M. & O'Hare, G.M.P. (1990)**, "GARP: A Rapid Prototyping Tool for Distributed Knowledge-Based System," Technical Report AI-92-2 (July 1992), Department of Computation, UMIST University, Manchester.
- [7] **O'Hare, G.M.P., Reddy, M. & Jones, A. (1992)**, "AMNESIA – Implementing a Distributed Knowledge-Based System using RAPIDO," in 'Proceedings of Expert Systems '92, 12<sup>th</sup> Annual Conference of the British Computer Society specialist Group on Expert Systems,' (Cambridge, December 1992), Cambridge University Press.
- [8] **Reddy, M. & Moon, J.N.J. (1995)**, "Development and Evaluation of Multi-Agent and Agent-Based Simulation Environments," 'DIMAS '95, Proceedings of the 1<sup>st</sup> International Workshop on Decentralized Intelligent Multi Agent Systems,' (Krakow, November 1995), pp393-401.
- [9] **Reddy, M. & Fletcher, G.P. (1997)**, "Intelligent Control of Dynamic Caching Strategies for Web Servers and Clients," in 'WebNet '97, Proceedings of the 2<sup>nd</sup> World Conference of the WWW, Internet, & Intranet,' (Canada, November 1997), pp440-446.
- [10] **Reddy, M. & Fletcher, G.P. (1998)**, "An Adaptive Mechanism for Web Browser Cache Management," IEEE Internet Computing, Vol. 2, No. 1, (January-February 1998), pp78-81.
- [11] **Reddy, M. & Fletcher, G.P. (1998)**, "Intelligent adaptive web caching using document life histories: A comparison with existing management techniques," in



‘Proceedings of the 3<sup>rd</sup> International Workshop on WWW Caching,’ June 15-17<sup>th</sup>, 1998, Manchester, UK.

- [12] **Reddy, M. & Fletcher, G.P. (1998)**, “Exp1: a comparison between a simple adaptive caching agent using document life histories and existing cache techniques,” in ‘Computer Networks and ISDN Systems,’ Vol. 30, Nos. 22-23, (November 1998), pp2149-2153.

### **1.3 Author's role in the work**

For papers [1], [2] & [6], the author produced all the preparatory work, literature surveys and initial drafts. O’Hare’s role was primarily for final preparation of the publications, involving proof reading, and suggestions for improvements. Papers [3], [4] & [5] were produced solely by the author.

For paper [7], much of the preparatory work was shared between the author and Jones.

Literature surveys and initial drafts were produced by the author. However, Sections 3, 5 & 6.1 and Appendix 1, were adapted by the author from Jones’ MSc dissertation report.

O’Hare’s role was primarily for preparation of the final publication, and to present the work at conference, due to an extended period of illness for the author.

Paper [8] was jointly authored by the author and Moon. The majority of the work was prepared by the author. However, Section 4, and parts of Section 5 were written by Moon.

Papers [9], [10], [11] & [12] were jointly authored by the author and Fletcher. Initial development of the simulation, and initial analysis of results, were shared between the author and Fletcher. The literature survey, proposal of the technique and experimental design were prepared by the author. All drafts of the publications were produced by the author, with Fletcher proof reading, and making suggestions for improvements.

This overview document now considers the two projects in detail: Section 2.0 presents RAPIDO, a toolkit for developing MAS applications by rapid prototyping; WebAgent, the second project described in Section 3.0, is a simulation testbed for performing experiments in improving Internet network performance. Both projects are set into the context of their

specific problems, before a rationale for the work is presented. This is followed by a description of the work achieved, and further described by reference to the documents in the portfolio. Each section is then concluded, with an evaluation of the work done and recommendations for further work. Finally, Section 4.0 describes how the two projects are related, and outlines the contribution to knowledge and future directions of research.

## **2.0 RAPIDO**

This chapter describes a toolkit for prototyping MAS applications, developed at UMIST under a SERC studentship between 1989 and 1992, and some closely related work performed at Glamorgan University in 1995. Section 2.1 provides an introduction to the specific problem area of developing and evaluating MAS applications. Section 2.2 describes the rationale for RAPIDO, and outlines the project objectives. The work done during the project is detailed in Section 2.3 and further described by reference to the portfolio papers, in Section 2.4. Finally, Section 2.5 discusses the conclusions and directions for future work.

### **2.1 Introduction**

In the late 1980s, the advantages of a Distributed Artificial Intelligence (DAI) approach to Knowledge Based Systems (KBS) were becoming recognised, and a plethora of DAI paradigms had been proposed. However, none of these models were capable of supporting problem solving in all areas; each being suited to particular domain types. A truly generic model of multi-agent architectures, combining all the separate paradigms into one universal representation was considered unfeasible at the time, because each paradigm displayed unique features. Nevertheless, they did show similarities in the requirements of their agents.

The design of intelligent agents has been identified as an important research direction within multi-agent systems (MAS) and identifying appropriate architectures is fundamental. Agent applications should allow many different co-existing components with complex interactions. These involve asynchronous parallelism, which requires process synchronisation and protection of internal resources; a problem common in real-time, concurrent and distributed systems. The construction of DAI systems would be facilitated by adopting a prototyping approach for evaluating the most suitable paradigm for a particular domain. Therefore, the design of MAS must be supported by providing tools and methodologies for developing computational implementations of agents and their communications.

The following section examines the motivation for RAPIDO, in light of previous work, and identify the research questions, which were subsequently raised. Finally, the objectives for the project are described.

## **2.2 Rationale for RAPIDO**

It was clear that tools were needed to aid both the acquisition and design of MAS prototypes, and help in the evaluation of different implementations of these systems under a number of DAI paradigms. It had been identified that a generic approach might allow more objective evaluation of implementation strategies in order to determine which paradigm to adopt, how to build it and how to partition the domain knowledge. A secondary concern was the question of how to implement real-time execution of and communication between agents. It was felt that the link between the parallel development of DAI and concurrent, distributed and real-time systems warranted further examination.

The main focus for developing RAPIDO was, therefore, to explore the effectiveness of using a pragmatic, generic model of agents as black boxes, linked by simple communication protocols, regardless of the underlying paradigm. Whereas many tools support a single paradigm and/or implementation method, the aim of the RAPIDO toolkit was to explore a variety of paradigms and to study performance under a number of different architectures. It was intended that the use of a layered architecture would allow different paradigms to be constructed in library form, with the intent that these libraries would only be accessible through generic tools to hide low-level details behind layers of abstraction, so that agent architectures could be created generically.

Comparison of the performance of different agent architectures may only be possible when experiments are controlled and repeatable [Hanks et al (1993)]. It was hoped that some degree of evaluation of the performance of different paradigms for a particular domain might be possible. In order to evaluate RAPIDO's ability to do this, a MAS application based upon a real-world case study, would need to be implemented under a number of paradigms. While this would not necessarily prove that the specific domain was best suited to a particular paradigm, it would provide a proof of the concept that the toolkit might be used in such an investigation. Therefore, the following objectives were identified:

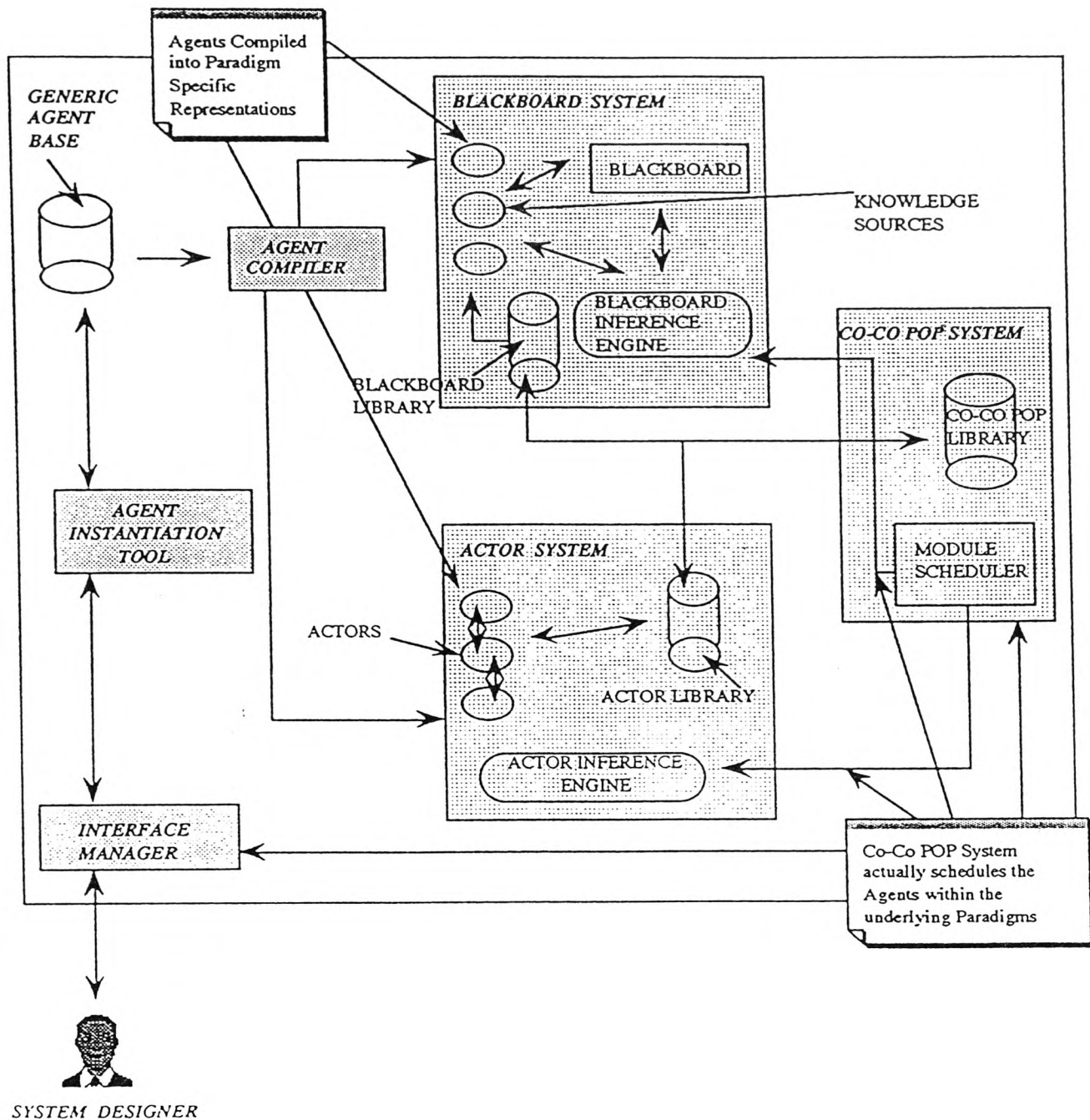
- ❖ A survey of existing DAI paradigms and applications, with an emphasis upon the underlying communication requirements;

- ❖ Examination of techniques in concurrent, real-time and distributed systems for the partitioning and communication between distinct processes;
- ❖ Design of a set of communication primitives supporting these techniques in a form accessible to an agent-oriented development tool;
- ❖ Implementation and comparison of several DAI paradigms using these low-level communications primitives;
- ❖ Design of some development tools to support construction of MAS applications via rapid prototyping techniques;
- ❖ Evaluation of the final toolkit by the development of a real application from a non-trivial case study.

### **2.3 Work done with RAPIDO**

In this section, the implementation of RAPIDO is discussed. This architecture has several layers of abstraction and an acquisition/representation language for the development of MAS prototypes; see Fig. 2.3.1. The first and lowest layer of abstraction is the CoCo-POP Testbed. CoCo-POP was implemented in Pop-11, using the low-level threaded process primitives to build up an extensive set of facilities for implementing concurrent applications. This provides Pop-11 with a library of objects capable of running in parallel, and a scheduler for overseeing their execution. Communication primitives allow for a range of protocols inspired by real-time systems research (including semaphores, monitors, remote procedure calls (RPCs) and ADA-style rendezvous process synchronisation).

Pop-11 process control primitives were used to implement co-operative multi-tasking and internal parallelism; module tasks can be interrupted by higher priority jobs before resuming. The scheduler can be configured to run in a round robin or priority-based manner, with priorities set by the agents themselves or by a scheduling agent; the default is for basic modules to perform this task, to allow for trace/debugging. Differing agent process requirements can be compensated for by scheduling slower agents with a greater priority.



**Figure 2.3.1 – The architecture of RAPIDO**

CoCo-Pop is able to execute heterogeneous agent architectures, but the agents themselves must handle translation of messages internally. Higher-level communication protocols, such as KQML, were not implemented at this level because they were paradigm specific. Therefore, the agent modules must contain their own translator/interpreters or use intermediary agents to act as go-betweens.

The second layer of abstraction consists of libraries of extended CoCo-POP modules or agent shells, rather like expert system shells but with added communication facilities. These libraries are object-oriented in approach as they consist of classes of modules with inherited generic slots that can be extended by the user. The use of a set of paradigm libraries allows us to avoid adopting a potentially inappropriate technique until evaluation of different approaches has enabled us to make an informed decision.

Two such libraries exist: Black Board Builder (BBB) and CoCo-POP Actor Simulation Testbed (CAST), which implement blackboard and actor based systems respectively. These were chosen because of their widely differing approaches to internal representation of knowledge, communications and methodologies: “Objects [Actors] have to know what to remember. Blackboards have to know what to forget.” [INSIGHT (1986)].

The final layer of abstraction is the RAPIDO prototyping tool, which uses a generic acquisition/design interface to make the underlying architecture less explicit. This tool was created to integrate between the separate layers, and implements a generic lifecycle for developing agent systems. RAPIDO can, therefore, be used to build DAI systems quickly from “off-the-peg” components; a process which also serves to shield the developer from the physical details of implementation. CoCo-POP assists this manner of incremental development by providing a set of pre-defined building blocks that represent generic agents.

The RAPIDO development lifecycle consists of several distinct stages: Define the names, internal and external skills of agents; lists of acquainted agents and which of their available skills may be called upon; and the rule set for each skill. For undefined skills, with the feedback required from the User agent, the appropriate interface would then be developed manually. See Figs. 2.3.2-2.3.5 for details of the RAPIDO interface.

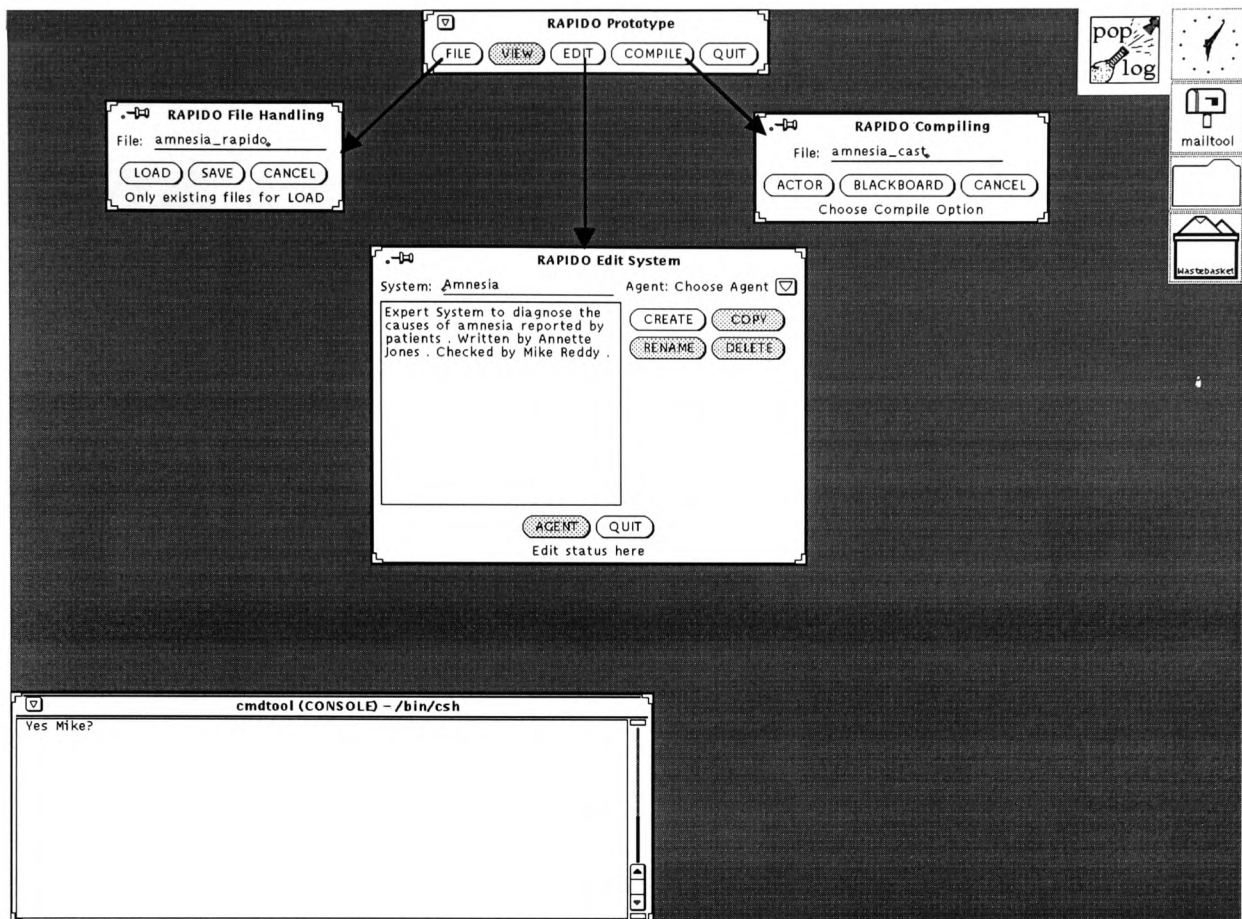


Figure 2.3.2 – RAPIDO top-level interface (Creating AMNESIA)

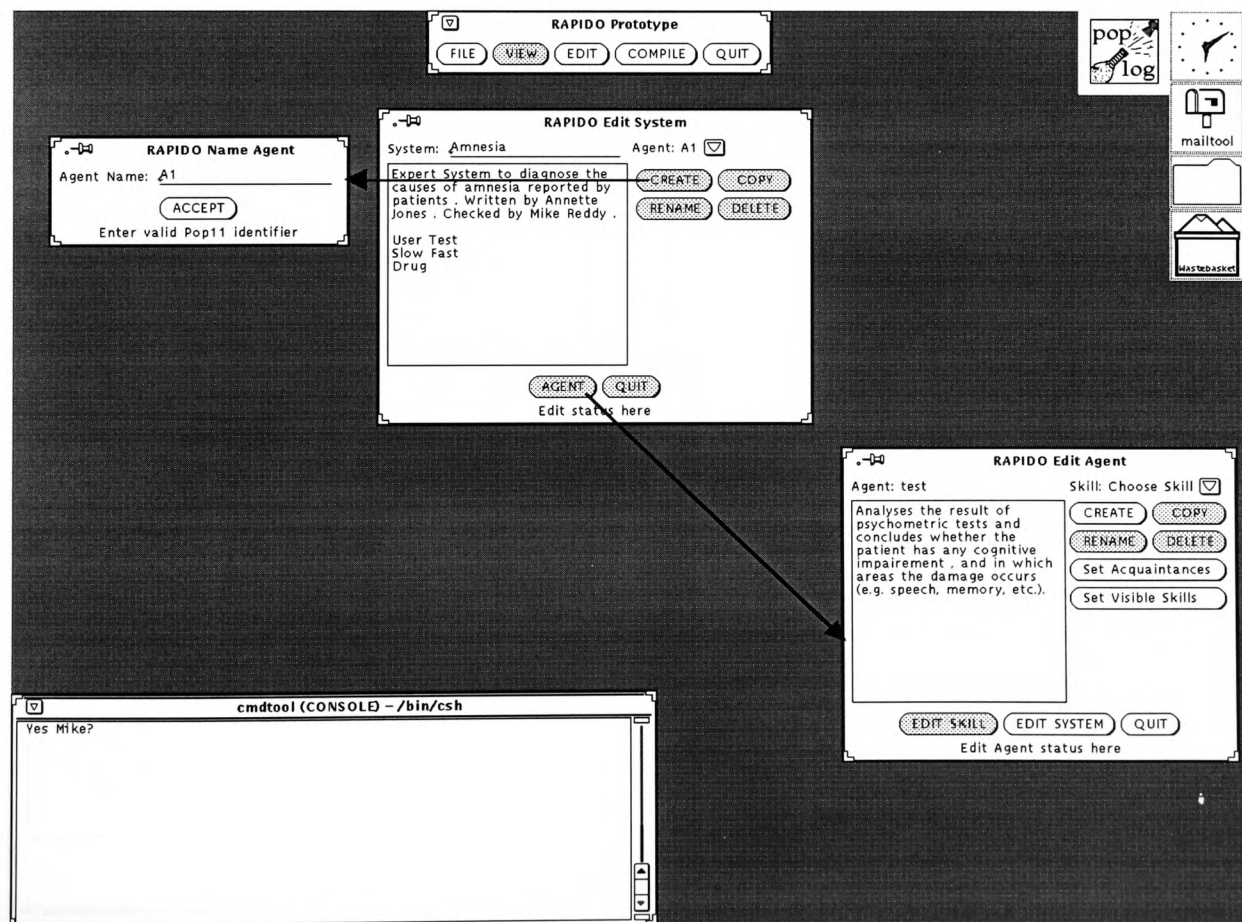


Figure 2.3.3 – RAPIDO agent interface (Creating the test agent)



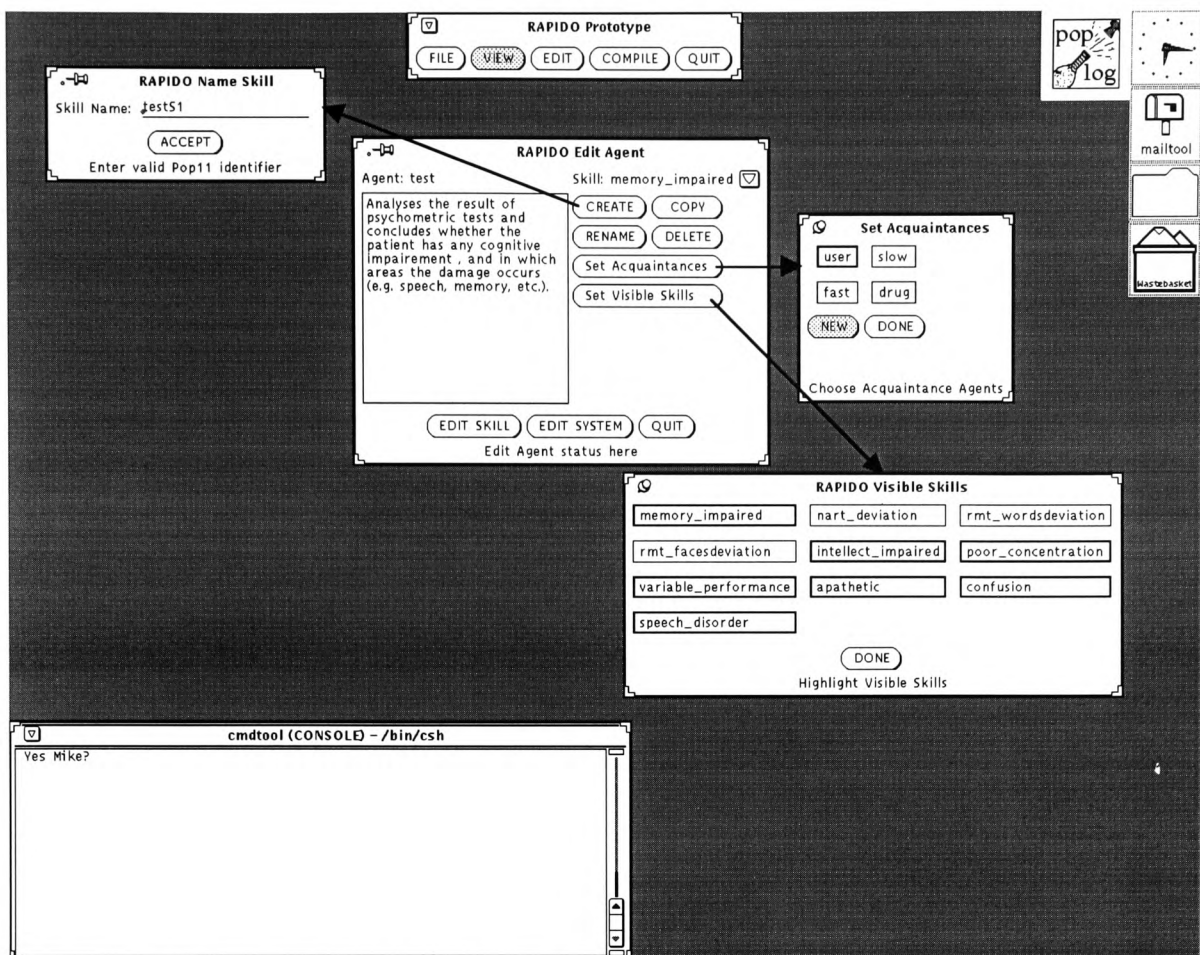


Figure 2.3.4 – RAPIDO skill interface (Creating and defining visible skills)

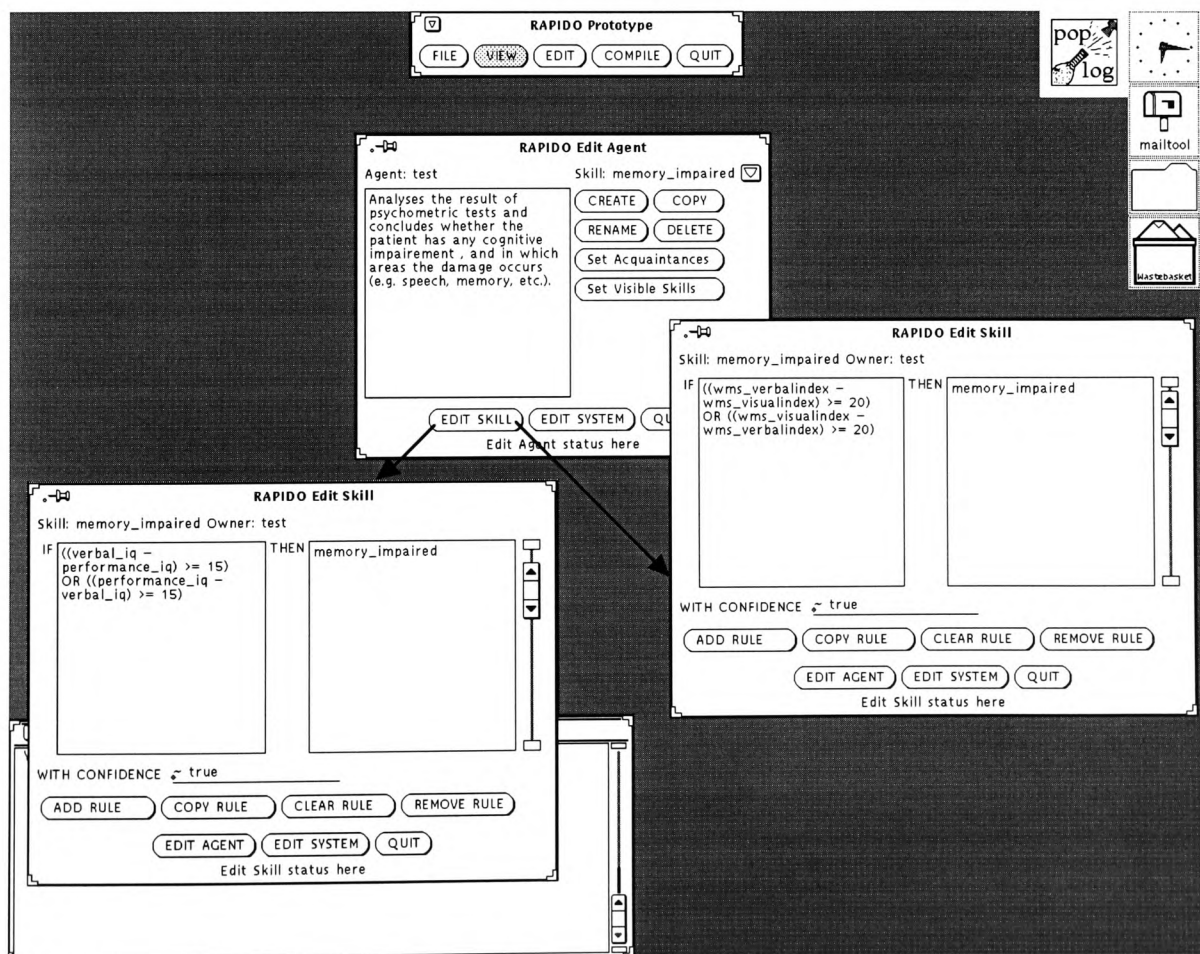


Figure 2.3.5 – RAPIDO rule interface (Setting rules for memory\_impaired)

## **2.4 Description of the RAPIDO papers**

The first publication in this section details the initial literature survey that inspired the RAPIDO project. The following technical reports describe the chronological development of RAPIDO in more detail. The penultimate publication describes the application of RAPIDO to a case study. Finally, the last paper evaluates the facilities provided within RAPIDO in terms of other related simulation testbeds and toolkits.

### **2.4.1 Paper 1: “The blackboard model: a survey of its application”**

**Abstract:** The need for co-operation and communication between knowledge-based systems (KBSs) has prompted research into the field of distributed artificial intelligence (DAI). A number of paradigms have been proposed – including the blackboard model.

A de facto blackboard model is described which contains three components: the blackboard data structure, knowledge sources and a means for control. To enable comparison between existing applications, a set of attributes has been distilled from the model. Identification of three distinct groupings of current systems has led to the proposal of taxonomy of blackboard systems. This consists of three generations of development: dedicated systems, generic shells and tool-based architectures. In light of this, an evaluation of the blackboard model is made, with respect to its significance to the field of DAI research.

### **Conclusions of the paper**

A survey of blackboard system implementations has revealed a generation-based taxonomy of development; these generations being approximately chronological and categorising the underlying philosophy of design. This model may also be valid in the more general framework of all DAI systems, but this has yet to be verified. Investigation of DAI applications is to continue in an attempt to evaluate whether the three-generation model can be usefully extended into a general taxonomy of DAI systems. The high communications content of DAI has also prompted an examination of Software Engineering techniques for modular design, high cohesiveness and loose coupling.

## **Commentary**

Although this paper was published in early 1991, the final draft was submitted in December 1989, and so is placed in this section first, to better reflect the chronology of development.

Due to the maturity of the blackboard model and the large number of implementations, it was decided to perform a survey of existing documented applications of the blackboard model as part of the research into DAI paradigms. A taxonomy showing the evolutionary development of DAI applications of the Blackboard model was identified. Several citations and requests for reprints of the paper were made to the author; see accompanying correspondence. The results formed the basis for initial development of the CoCo-POP architecture.

### **2.4.2 Paper 2: “CoCo-POP A Development Testbed for prototyping Distributed Knowledge-Based Systems**

**Abstract:** The advantage of a multi-agent approach to knowledge-based systems research are well documented. However, there are problems in developing Distributed Knowledge-Based Systems (DKBSs): which paradigm to adopt, and how the system should be implemented. Building a prototype application would enable the assessment of the most suitable module for a particular domain. An environment for the speedy construction of multi-agent systems from pre-defined components would facilitate this process.

CoCo-POP, a testbed for building prototype multi-agent systems is described which provides a store of low-level primitives, a construction interface and libraries of user configurable components for developing ‘off the peg’ prototypes. An example blackboard system is then described to demonstrate how an application may be achieved more easily using CoCo-POP.

It is proposed that such a tool-based approach to the development of multi-agent systems is vital for the evolution of ‘real world’ applications. It also serves to emphasise the need for cross-fertilisation from the neighbouring fields of concurrent and distributed systems.

### **Conclusions of the paper**

The communications requirements of DAI paradigms were investigated and compared with similar techniques from concurrent and real-time systems. Commonalities were identified, which suggested that recent developments in real-time systems might be used within a DAI

tool. The architecture of CoCo-POP, a testbed for implementing parallel and distributed agents was the defined. A set of data definitions and procedures was implemented in Pop-11, which enabled pseudo-concurrency of a module data type, and remote procedure calls and message passing via a mail system were supported to provide basic communication protocols. Test programs such as the Producer Consumer Problem were used to check that non-determinism was possible.

### **Commentary**

This technical report was the result of the first 18 months of practical work at UMIST, and shows that, although much of the coverage was low level, there was (from its earliest conception) recognition of the need for abstraction and support for the design of DAI systems. The blackboard model was also implemented under CoCo-POP, as a library called Generic Blackboard Shell (GBS). This development ran in parallel with that of CoCo-POP, and was not documented in detail at the time. The name of this library was later changed to Black Board Builder (BBB), as GBS was discovered to be already be in use.

### **2.4.3 Paper 3: “A Proposal for an ACTOR-Based Extension to CoCo-POP – CAST”**

**Synopsis:** This report will outline the methods of representing and implementing ACTORS within the CoCo-POP Testbed. Firstly, the script definition language used for Actor systems will be described. This is followed by a discussion of how this language may be used to create actors. Actor structure is then defined in terms of CoCo-POP modules; the components required will be described. Finally, the control mechanism within and between actor modules is declared.

### **Conclusions of the paper**

The Actor model was analysed in terms of its control and communications requirements. The case for using Actors was made and an Actor library was proposed as an extension to CoCo-POP. Finally, a timetable for development of CAST was described.

### **Commentary**

This use of the implementation of actors within CoCo-POP was a deliberate tactic for driving its development. By picking an architecture that was diametrically opposed to that of

blackboard systems, an attempt was made to ensure that the structure was generic, yet useful, and flexible enough to represent different DAI paradigms.

#### **2.4.4 Paper 4: “CAST – Progress for the Implementation of Actor Communications”**

**Abstract:** Actor communications are often complex and difficult to represent well when standard request-reply format is not applicable. This report discusses the problem of widow requests and orphan replies within the actor syntax implemented in CAST. A summary of the widows and orphans follows, with an account of how the problem is to be solved using an extension to the existing CoCo-POP actor module shell.

##### **Conclusions of the paper**

This paper described the problems with simple actor-based message passing, when transactions were interrupted, or not completed. When a transaction is redirected or deferred, it introduces novel problems for the implementation of actors. In particular, the problem of widows and orphans were discussed. A proposal is made for using tags for identifying return addresses. Finally, EBNF and pseudo-code for the proposed solution were outlined.

##### **Commentary**

The issue of implementing DAI applications is related strongly to communications, which can benefit from cross-fertilisation with the fields of Computer Science to avoid re-inventing the wheel. In order to realise this model of Actor communications required an extension to the features offered by CoCo-POP. This shows the wisdom of using different paradigms to drive the evolution of CoCo-POP.

#### **2.4.5 Paper 5: “CoCo-POP Generic System Capture”**

**Abstract:** CoCo-POP is now at the stage where two paradigms, for blackboards and actors, have been implemented in library form; these are called B3 (i.e. BBB) and CAST respectively. Discussion must now turn to the manner in which users may create distributed problem solvers (DPS) using these constructs.

This discussion document will outline the method by which the structure of an application is taken from the user and transformed into a CoCo-POP system. A model of a possible user dialog is described, with reference to the information needed to define an application at the

system and agent levels. Treatment is graphical, with some supporting text to explain the procedures.

### **Conclusions of the paper**

This paper outlines the initial development of a generic model of DAI systems, which consists of a generic model of inter-agent dependencies is based upon a representation of agents, skills, requirements and acquaintances.

### **Commentary**

The proposed model was focussed heavily upon the communication requirements, inspired by modular decomposition common in Software Engineering. From a historical perspective, this model shares many features with early object-oriented techniques. However, there was still the need for a design interface that could represent agent applications without being tied to a specific paradigm.

#### **2.4.6 Paper 6: “GARP: A Rapid Prototyping Tool for Distributed Knowledge-Based System”**

**Abstract:** This paper will outline a Distributed Artificial Intelligence (DAI) toolkit which enables the rapid prototyping of Distributed Knowledge-Based Systems (DKBSs) for a variety of paradigms – currently blackboard and actor-based systems – This approach is based upon capturing the requirements of the developer by the use of a generic model of agents. This specification is then transformed into paradigm-specific applications for demonstrating first-stage prototypes.

The toolkit is modular in structure, with tools built from low-level primitives in increasing orders of abstraction. Facilities exist for viewing and animating the features of these paradigm-specific prototypes. Further work will aim to increase the robustness of these tools, and provide further facilities for evaluation purposes. Other areas for improvement include the generic handling of interface requirements, and evaluation tools for determining the most appropriate paradigm for particular domains.

### **Conclusions of the paper**

The need for development tools for DAI systems is identified, especially for prototyping. A generic model of DAI is proposed, which is based upon control and communication

requirements of agents. The ability for this model to be compiled under a number of DAI paradigms is described in detail. Implementation of these paradigm libraries, and the underlying execution environment are then outlined. Finally, the pros and cons of this approach are described briefly.

### **Commentary**

This paper is the first in the portfolio to recognise the term ‘multi-agent’, which was becoming more representative of research directions in DAI. Note: GARP was an acronym used as a working title for the toolkit. Later on, it was felt that there was not sufficient theoretical underpinning to make any claim for the theoretical usefulness of the model, which was pragmatic in nature. Therefore, in later work, the system was renamed Rapid Agent Prototyping DOrain (RAPIDO), due (in part) to the need to emphasise the rapid prototyping aspect, rather than the generic agent model.

#### **2.4.7 Paper 7: “AMNESIA – Implementing a Distributed Knowledge-Based System using RAPIDO”**

**Synopsis:** This paper describes the design, implementation and testing of a Distributed Knowledge Based System (DKBS) called AMNESIA. AMNESIA seeks to diagnose memory related disorder illnesses. The system was developed using RAPIDO (Rapid, Agent Prototyping DOrain) a rapid agent prototyping environment. The development of AMNESIA was useful in two senses: Firstly, it served as a case study in the design and implementation of a multi-agent system. Secondly, it acted as a medium for the testing and driving of RAPIDO. RAPIDO is a state-of-the-art tool whose usefulness and usability needs to be assessed, and this application provided a perfect opportunity.

### **Conclusions of the paper**

The need for toolkits to support the Knowledge Engineering lifecycle for DAI applications was described. The RAPIDO toolkit was applied to the problem domain of cognitive malfunction, and a DAI system, AMNESIA, was developed. The case study went some way to justifying the generic agent model, and the tool-based approach to design. However, the need for further tools for monitoring and evaluating agent performance were identified.

## **Commentary**

One of the issues raised by this paper was the need for further evaluation metrics and tools for comparing the performance of different paradigms. However, the paper's emphasis upon the case study was a deliberate attempt to claim that real MAS applications could be constructed with RAPIDO; that it could be applied to complex, real world problems. It was important that this should be held up to scrutiny, as many papers described the efficacy of various toolkits and testbeds, but only used simple 'toy' examples to validate their existence.

### **2.4.8 Paper 8: "Development and Evaluation of Multi-Agent and Agent-Based Simulation Environments"**

**Abstract:** Testbeds are available for experimenting upon multi-agent systems (MAS).

However, there is no agreed design methodology or set of metrics for evaluating agent architectures. Determining the most appropriate paradigm, or the best distribution of knowledge in heterogeneous agent systems, is not a trivial task. This is made more difficult when performance measurements do not consider the underlying implementation. Available toolkits may also force developers to adopt inappropriate knowledge representation and implementation techniques.

This paper proposes a tentative set of metrics for evaluating MAS development tools and agent-based simulation environments (ABSE). These metrics are applied to two different MAS testbeds: RAPIDO, an agent development toolkit, written in C and POPLOG; and MARINES, an object-oriented ABSE developed using C++.

The application of the proposed metrics for evaluating MAS toolkits is then discussed. The results show the strengths and weaknesses of toolkits and ABSEs for developing MAS applications. Finally, suggestions are made for future use of simulation environments in the development of MAS applications.

### **Conclusions of the paper**

The case was made for MAS development toolkits and simulation testbeds for evaluation. Metrics for evaluating the facilities of such systems were proposed. These metrics were then applied to two MAS platforms: RAPIDO and MARINES, and their pros and cons were then



described. Finally, the case is made for examining the effect of low-level implementation on agent performance, especially for safety critical systems.

### **Commentary**

The DIMAS paper marks the distinct change of emphasis in the author's research from symbolic to 'emergent' representations. However, the paper raised the question: Were these proposed metrics useful for initial analysis of a problem domain to aid development of a agent-based simulation, as well as evaluating applications for taxonomic purposes? The development of these evaluation metrics were instrumental in the later development of the WebAgent testbed, described in Section 3.

## **2.5 Conclusions and Future Work**

It is clear that a generic model of DAI would be useful for developing DAI applications. An attempt has been made to provide such a model as far as pragmatically possible by treating agents as the basic components of all DAI paradigms. RAPIDO was developed to support exploration of implementation options for co-operative, communicating MAS applications. This tool-based approach is necessary for the development of future 'real world' applications, as it provides support for the developer without reference to the method of implementation. It also allows for cross-fertilisation from other areas of computer science, such as concurrent and real-time systems, by providing a model of DAI communications.

Many toolkits for developing multi-agent system applications or agent-based simulations require the acceptance of a particular agent architecture. SIM\_AGENT, a similar toolkit developed several years after RAPIDO, was an object-oriented toolkit for creating simulation environments for multi-agent systems [Sloman & Poli (1995)]. Agents could consist of a combination of rule-based or neural architectures, allowing both deliberate and reactive strategies. Parallelism was simulated using a round-robin scheduler using a double pass. The first pass was used for sensing the environment, receiving communications from other agents, and planning activities. The second was used for passing messages to other agents, performing physical actions and resolving conflicts.

This technique was used to overcome difficulties introduced by the order of scheduling agents under simulated parallelism. Simulated time was discrete only, rather than real-time to compensate for agents of radically different structures, which might be disadvantaged in simulation (e.g. neural networks would perform slower within SIM\_AGENT than if they were implemented within an optimal framework). SIM\_AGENT aimed for flexibility rather than optimum performance, or realism of simulations, being essentially a testbed for evaluating different heterogeneous architectures.

It is interesting to note that, while the parallel development shows some remarkable similarities, there are some features of RAPIDO which are superior to this later work; the representation of time and concurrency for example. Although dated by today's standards, having only the rudiments of Belief, Desires, Intentions (BDI), RAPIDO has features that are still relevant to present research into Agent-Based Simulation Environments (ABSE).

RAPIDO is a generic toolkit, flexible enough to allow rapid prototyping but with enough structure to support incremental development. It can be used for exploring different architectures as well as developing real-time multi-tasking MAS applications. Although performance metrics are limited, it is also possible to evaluate (empirically at least) the effect of implementing an application under a particular set of DAI paradigms. (An evaluation of RAPIDO, in terms of its 'agency' is made by Reddy & Moon (1995) [p397] in Appendix 1) One future improvement will be to devise performance criteria for prototype evaluation under a number of different paradigms. Future incarnations of RAPIDO will need to incorporate a browser agent to enable the developer to supervise the evaluation of application performance in a more quantitative manner.

### **3.0 WebAgent**

This section describes both the WebAgent simulation of the Internet, and Exp1, a simple adaptive agent for cache management. This research was performed at Glamorgan University between 1996 and 1998. Although the main emphasis of this report is upon agents, in Section 3.1, we discuss the caching problem domain in detail to set a context for the later work. Section 3.2 provides a rationale for the development of WebAgent and Exp1, and outlines the project objectives. Details of the development of WebAgent and the work done with Exp1 are explained in Section 3.3, and further described by reference to the portfolio papers in Section 3.4. Finally, Section 3.5 evaluates the performance of WebAgent as an experimental testbed, and Exp1 as a caching agent, before examining future research ideas.

#### **3.1 Introduction**

The Internet is a complex distributed environment, but techniques for network control (i.e. online and off-line monitoring and management of the network resources) are required to exploit new technologies. Caching benefits the users (faster access), network providers (less bandwidth) and content providers (reduced server load), but there are contradictions: users are selfish, wanting the most up-to-date documents. Service providers want to use stored (and possibly stale) documents to provide economies of scale. Meanwhile, content providers want accurate information on document usage, as their revenue may be based upon hit rates.

Caches have evolved through two generations, and are beginning a third: Proxy servers, focussing upon end users, and requiring manual configuration; Semi-transparent or hierarchical caches, aimed at supporting network providers, but requiring configuration by the system operators; and Fully transparent network caches, which do not require manual configuration by the end user or the network provider [INFOLIBRIA (1998)]. These systems involve pre-fetching and/or modelling of popular documents, with delayed or intelligent validation techniques.

Caching is completely distributed, with latency being quite high for restricted bandwidth areas (such as some commercial ISPs). Centralised cache hierarchies are considered to be

infeasible for large networks. Distributed hierarchies are autonomous, and are used almost exclusively on the Internet, but introduce further communication latencies. Cache maintenance is, therefore, a distributed, time-varying problem. However, the problem is unique, in that the distribution of document requests are, in the long term, most likely to be self-similar, rather than tending towards a lognormal curve. In fact, document popularity conforms to a Zipf distribution [Zipf (1949)]. Furthermore, document requests arriving at a server adhere to a Pareto distribution when the network approaches saturation, rather than the expected Poisson distribution [Cunha et al (1995)]. Certainly, in the short term at least, user preferences may introduce a significant dependence upon the documents. This might account for the normally low hit percentage for web caches (rarely above 50%). However, it is possible to make short term approximations of likely future behaviour for off-peak periods or less popular servers. Crovella & Bestavros (1995) have identified that self-similarity is not always present during off-peak hours. They attribute self-similarity to user behaviour and network architectures under saturation or near-saturation conditions.

It is commonly believed that smaller files should be cached over larger ones. This represents both the fact that more clients are satisfied when more files are stored, and that smaller files are more popular in accordance with Zipf's Law [Cunha et al (1995)]. An alternative strategy is to store larger files to give a large byte hit rate, but this does not satisfy the majority of clients, and is potentially inefficient<sup>1</sup>.

Web document requests and transmission times are stochastic and non-stationary; in that they are not completely random and not completely ordered: It is likely that a document will be requested by a lot of independent sources during the same time period but this behaviour can change quickly<sup>2</sup>. It will never be possible to have optimal caching without complete

---

<sup>1</sup> To prevent this, many caches use thresholds to determine the maximum size for cached documents.

<sup>2</sup> An example of this was relayed to me by the proxy server SysOp from Lancaster University, who noticed that the latest version of Netscape was suddenly being requested frequently, using excessive bandwidth, but

knowledge of the network. However, communication overheads may prohibit even a modicum of information about neighbours. Neither is it appropriate for direct communication between ‘peered’ caches, as is advocated by reciprocal caching strategies. The only legitimate means available for propagating information between caches are web documents requests. Therefore, it is impossible to know the status of the complete cache hierarchy.

Effectively, caches only have information about the local state and the past state of the original server, which will never be timely, as it is obtained by document requests propagating up the cache hierarchy. Static caching techniques make no consideration of current state of the network or actual use of documents. Adaptive caches can adapt to time and spatially varying traffic conditions, but can be prone to oscillation, causing fluctuations in performance.

Inconsistencies can arise when change in document usage is catastrophic; documents suddenly become popular, while other files are suddenly no longer of interest.

In order to assess the validity of various caching techniques, there are a number of different performance metrics: byte hit rate, document hit rate, and byte throughput are commonly proposed in the literature. However, these are rather arbitrary measures of cache performance, and none of them reflect ‘fair practice’, where end-users get a quality of service that does not impinge upon others, and/or guarantees better service for a higher premium. Neither do these approaches to cache management consider the network overhead (and to a lesser extent the overhead for retrieval from secondary storage). Most are focussed upon memory, bandwidth and computation costs. We have a conflict of interest between what are good metrics for the end-users and service providers.

On the one hand, the requirement is for the shortest delay, for which the cost function is based upon the delay in receiving up to date documents, irrespective of resources required to achieve that aim. For this case, the metric is time waiting for documents (including time

---

not being cached, due to its size exceeding the threshold. This threshold was manually increased for a short time, to allow requests to be served from the cache, rather than requiring a link to the server [Bennett (1998)].

waiting for acknowledgements that cached documents are not stale). Users require low cost, high quality, timely service, that is fault tolerant and reliable.

On the other hand, the providers seek to minimise traffic and infrastructure costs from a global perspective, for which the measure of success is not quality of service for users but cost effectiveness. In this case, these desires are constrained by the physical characteristics of the network, as well as the constraints of the network service providers, who are attempting to provide the best possible service for the least capital outlay to maximise profits.

These requests are contradictory, and at two extremes of the following axes: Storage costs 'v' Communications costs; Storage capacity 'v' Network Bandwidths; Download times, 'v' Document freshness, etc. Simply put, the problem of caching is time varying and stochastic, has to meet several objectives and must work within several orthogonal constraints. The distributed nature of the problem means that it has significant 'hidden' states, making it a classic reinforcement learning problem [Kaelbling et al (1996)]. However, network delays are remarkably difficult to analyse, and simple means of measuring performance are less useful than considering the empirical distribution, given the wide variance in the magnitude of network delays [Di Caro & Dorigo (1998)].

It has been suggested that frequency of use should be the sole arbiter of cache management. That way, the byte space in a limited cache is being used to the best possible extent. If this involves caching large (but popular) files, even when this means that less files are stored, it also clears bandwidth, so that requests which cannot be served from the cache have extra capacity, which otherwise would have been used retrieving the larger documents. This contradicts the popular technique of setting a threshold on document sizes and neatly avoids the difficulty in deciding the threshold level<sup>3</sup>.

---

<sup>3</sup> Thresholds often require manual setting and there is no clear guideline for what the threshold should be. The level is often determined arbitrarily by the SysOp for the cache server.

However, it is notoriously difficult to obtain information about document usage. Robustness to wrong estimates is needed because the Internet traffic is self-similar, and almost impossible to predict in the long term [Almeida & de Oliveira (1996); Arlitt & Williamson (1996)].

Although, if web traffic is self-similar, this means that the problem might be less complex than it appears. Provided that a strategy adapts quickly, and does not have too strong a historical perspective, it can still be used to predict behaviour.

Caches should match the changing nature of the Internet, by attempting to fulfil the agreed metrics for good performance, rather than being limited to pre-defined strategies. Neither should they improve performance for the few at the expense of the many; fairness is often absent in the existing environment, with the heaviest users getting the most bandwidth, at the expense of the lighter users. However, any constrained cache management technique – such as the commonly used thresholds [Williams (1996)] – is in some way deterministic and will, therefore, be less than optimal under all conditions [Singh et al (1994)]. Probabilistic approaches are more appropriate for incomplete information problems, such as predicting document usage. They can make use of previously learned, reliable information, so long as they are flexible enough, without being too retrospective.

In the following section, we examine the motivation for the WebAgent Testbed, and the Exp1 adaptive caching agent, in light of previous work, then identify the research questions, which were subsequently raised. Finally, the objectives for the project are described.

### **3.2 Rationale for WebAgent**

Internet traffic problems, such as cache maintenance, cannot afford the luxury of ‘high cost’ solutions provided by deliberative or planning agents, due to time constraints. However, these issues could be addressed effectively by reactive agents. So, it is likely that simple, adaptive agent techniques might also be used in this new and growing field. It has been recognised that probabilistic or statistical approaches might provide the basis for an effective strategy for cache management. The aim of developing Exp1 was, therefore, to investigate the validity of using exponential smoothing to predict document usage.

Experimental evaluation of such a novel approach is necessary, because theoretical models are inadequate for understanding real Internet traffic. However, the traditional approach, which is to use trace-based simulation, may not be appropriate, as it relies heavily upon server logs. These logs represent the access patterns of many users and, therefore, only allow general evaluation of cache performance. Client logs would allow individual performance metrics, such as perceived retrieval time, but access to a statistically large enough sample is unlikely, and might be prone to geographical and temporal peculiarities.

Any Internet simulation should allow us to evaluate both end user, as well as global performance. This can only be done by the simulation of client requests rather than merely using trace-based simulations, which do not represent individual user access patterns. Clearly, this is most appropriate when evaluating client browser caching strategies, which would be poorly represented by a trace-based technique. Furthermore, there is a need for quantifiable, configurable experiments, showing realistic document dynamics (including document modification, as well as document requests). A 'real-time' simulation would also have the benefit of repeatability, allowing different caching techniques to be measured and compared accurately.

Therefore, the aim of developing WebAgent was to facilitate the testing of adaptive agents for cache management to reduce network traffic, while monitoring their timeliness. The use of an agent-based approach would allow a continuous, but repeatable simulation of the Internet, which would be impossible with finer grained, traditional network simulations (i.e. packet/node based simulators) or trace based approaches. Therefore, the following objectives were identified:

- ❖ Examination of network performance of web servers, clients and caches, both directly from local server logs and generally available public data;
- ❖ Design of an agent-based simulation testbed, based upon the server statistics, to represent the internet;



- ❖ A survey of existing caching algorithms, with an emphasis on adaptive techniques or agent-based approaches;
- ❖ Implementation of several existing cache algorithms to provide a comparison with the agent based approach.
- ❖ Development of an agent-based technique for cache management and evaluation of the proposed intelligent adaptive caching agent.

### **3.3 Work done with WebAgent**

Exp1 is a simple adaptive agent, which implements a novel approach for predicting document usage in communication networks. The model is adaptive, and is based upon document life histories computed over time. For each document an estimated mean time between requests provides an empirical measure of the request frequency, and, therefore, the value of caching a document. The basic strategy follows four guidelines: (i) models should be updated at regular intervals, and in parallel with data traffic, with little or no additional computational cost; (ii) models should be relative, to simplify comparison between cached documents; (iii) models of document usage should be reinforced from historical, as well as current behaviour; (iv) models should not follow all changes in document usage, but depend upon short term estimates only. This is the recency 'v' frequency debate, which has previously been heavily weighted to the former approach, because there has always been a difficulty with accurately determining the latter.

The technique used within Exp1 was exponential smoothing, which makes use of local stochastic data, using past transactions to build up an incremental model of document usage, in order to direct future behaviour. The sampled mean provides a measure of the request frequency, rather than merely the recency, as is true for the popular Least Recently Used

(LRU) technique. This is the key benefit of Exp1 over LRU, as it prevents oscillations in the prediction of document usage<sup>4</sup>.

In order to validate this approach, a realistic simulation of the Internet was required to enable comparisons between static and adaptive caching algorithms. The design of a meaningful testbed to compare competing algorithms was no easy task; cache hierarchies are governed by many factors, which may interact in non-linear and unpredictable ways. An agent-based approach was chosen to minimise the number of interacting components to allow the caching components to be evaluated in isolation without sacrificing realism. Therefore, a limited set of criteria were modifiable in the experiment: cache size; network performance; request rates. The WebAgent simulator was developed to closely mirror the essential features of web document transactions on a generic communication network, consisting of several document providers and consumers operating under the same conditions. The simulation focused on a conceptual network, built upon spatial and temporal traffic distribution patterns, rather than network characteristics and low-level communications protocols. The removal of simulating the physical attributes of a network allowed initial development of WebAgent to be implemented in PROLOG<sup>5</sup>.

---

<sup>4</sup> Further details are available in Section 3.4. However, Exp1 balances the recency and frequency measurements of a document using a weighting,  $\alpha$ . With  $\alpha = 0.1$ , the previous 22 transactions make up the 90<sup>th</sup> percentile. This is to ensure that the algorithm is not too retrospective, nor too reactive.

<sup>5</sup> PROLOG is not a typical implementation language for network simulation, but did allow for rapid development, and provided extensive list handling primitives. Clearly, a language such as C++ would have been quicker for simulator runs, but much longer in the initial development stage before useful results could be obtained. However, PROLOG was computationally efficient, given that we were modelling conceptual rather than physical networks, allowing for ease of modification for particular experiments without sacrificing the validity and scalability of the results.

Document transmissions were represented as complete transactions, rather than at the level of individual packets, with transmission delays and breakdowns being simulated by an accumulated transmission cost, based upon an aggregation of performance across gateway nodes. There was no attempt to represent a real transport layer, with network errors and congestion control, as each additional layer of complexity would have a considerable impact upon simulator performance. Furthermore, interactions within underlying layers would not have contributed to the evaluation and comparison of each caching algorithm.

Finally, the simulator model needed to represent a typical internet infrastructure, with no guaranteed quality of service, rather than a connection-oriented network, that makes bespoke, semi-permanent connections to maintain a minimum level of service. The Internet adopts a hierarchical Autonomous System approach [Moy (1998)]. WebAgent extends this metaphor to represent sub-networks as single 'super nodes' connected to each other through gateway or routing nodes. These super nodes consist of local users and suppliers of information (effectively an Intranet), who may also make external requests and supply external demands from outside the logical topology of the local network.

Super nodes can then be considered to both supply and demand services from the Internet; the exact balance between supply and demand depending upon the organisation. For example, an ISP (Internet Service Provider) will have thousands of users making requests for web documents, but may also have to offer caching to both reduce access time for popular documents and to reduce the bandwidth requirements and corresponding capital outlay. ISPs may also provide extra services (e.g. AOL, CompuServe) for commercial reasons, as well as to encourage this concept of internal features for recruitment purposes. These extra features might also, ironically, increase hardware expenditure by increasing the external traffic into the node.

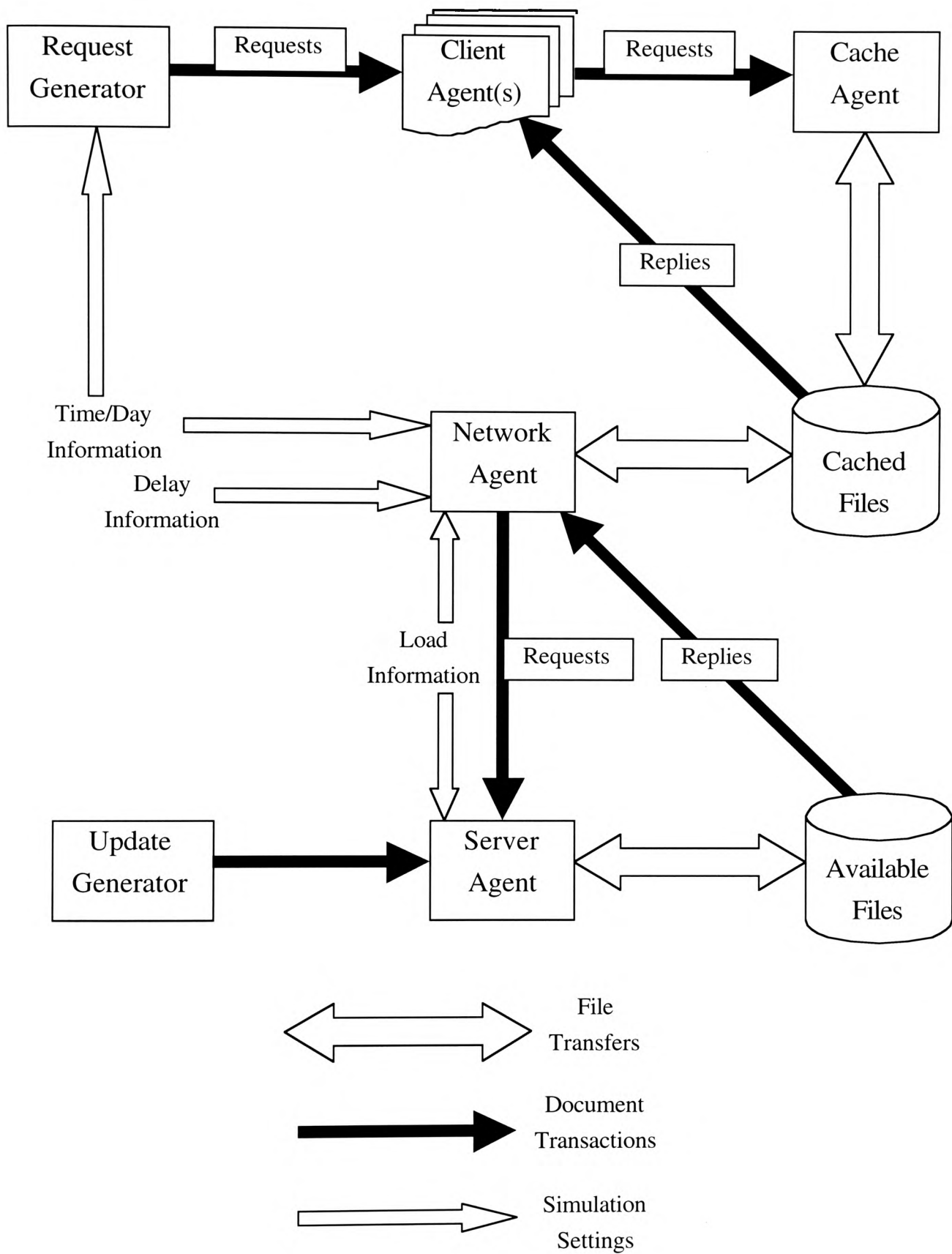
WebAgent was, therefore, implemented as a set of distributed super nodes: non-mobile agents communicating only implicitly via document transactions. (See Fig. 3.3.1.) As the aim

of the simulation was to evaluate performance for a number of caching algorithms, the Server Agent was implemented to represent document sources, while Client Agents represented several 'sinks' (i.e. client browsers, or proxy servers), operating different caching algorithms. Each Client Agent requested the same documents, chosen by the Request Agent, and encountered the same network performance, set by the Network Load Agent. Differences in efficiency were, therefore, based entirely upon whether the Agents had retained the document in their caches, thus saving download time and bandwidth.

An upper limit of simultaneous transfers, or fixed bandwidth, was not enforced within WebAgent, as this would have required packet level simulation. This was compensated for by the assumption that peak loads would have much larger download times. This factor for peak times makes the WebAgent simulator unique, in that web traffic was generated using a pseudo-random process, rather than relying upon trace data. It is a discrete event simulator using a pseudo-random generator (to allow repetition of specific conditions) to generate the event list. This allows the requirements and constraints upon the simulation environment to be modified in ways that could not be done using a trace-based snapshot, provided by server logs alone. The statistical generation of cache request data was based upon analysis of network performance for several large proxy servers as well as the http and ftp logs for a local server at the University of Glamorgan. Given that the domain for the experiment was web document caching, rather than network routing, it was deemed acceptable to avoid using a fine-grained representation. This is not unusual, as most other simulators are based upon trace-based data generation, which can only be large-grained; considering document hit rates, with only a meagre attempt at modelling byte transfer or actual network performance<sup>6</sup>.

---

<sup>6</sup> Often, researchers make use of caches with 'infinite' storage, for the sake of convenience, which prejudices the need for realism in simulation. This is because their work is looking at correctly predicting which files to cache, irrespective of whether these algorithms could actually be implemented or work in practice.



**Figure 3.3.1 – Structure of the WebAgent Testbed**

WebAgent retains many of the basic components of a real routing system: servers, gateways, users and providers, modelled as 'super-nodes', with differing capacities and bandwidth requirements. This conforms to the idea that network nodes can be represented simply, using only three parameters: capacity, latency (loaded and unloaded) and loading [Perkins & Machin (1997)]. The model is simple, but captures key features, such as the operational characteristics of network and user behaviour, while allowing us to represent different scenarios in a reproducible way. The limited resources for transmission were reflected in the generated rate of document requests and the download time for documents, based upon trace logs for the University web server. Generating download times is made more difficult because a distribution curve is 'heavy tailed' (a feature of self-similarity). For the sake of simplicity, given that all cache approaches were to experience the same network performance, a lognormal distribution algorithm was used to produce randomly variable download times (i.e. network performance varied around some mean, with a maximum value set to the bandwidth of a typical internet connection).

For simplicity, document download time was determined by a fixed cost (latency and load) and document size. It was assumed that each packet travels the same route and experienced the same delay in transmission. The simulation of time was implemented using discrete time units, conforming to the number of seconds between transactions at high peak periods for a web server. The actual traffic analysis approximated to 2-3 seconds between transactions at peak times for the Glamorgan University web server. Clearly, at off peak times, the gap between transactions could be significantly higher. On average, there was about ten seconds between transactions. This defined the maximum and minimum number of potential transactions the server would receive, over 14,000 requests per day. This might be considered a light loading, compared with more popular servers, but does not take into account internal access from the campus intranet, which was excluded from the analysis. The emphasis on external traffic was due to the need to stretch the efficiency of the cache; the inclusion of internal traffic would have skewed the focus of the data unfairly towards fewer documents, making caching decisions much more straight-forward.

Parallel requests between different simulated clients and servers, were batched up together. However the network performance for each request was calculated separately. Random spatial factors were introduced into the pre-determined value for maximum bandwidth, as well as temporal ones (e.g. time of day, day of week). The simulator would often generate bursty performance, due to the accurate representation of the changing interest in documents, based upon time of day and of trend and fad influences. The corresponding bit rate was then used for the whole document, and (it was hoped) would represent a wide class of possible patterns that could arise in most network traffic situations. Smaller time steps, of less than a second might have increased realism, but at the expense of slowing the simulation without improving the results. It is more important to capture the minimum time period between critical events. The actual rate of use of documents was stored for each individual file and was modified over time to simulate changing interest in documents. Document download times were also represented as a number of time steps, allowing the clients to know when a transaction was completed, by checking the global clock. Therefore, a simple model of network performance in the context of web document transactions, clients, servers and caches has been proposed. This abstract approach both simplifies what can be a complex architecture, and eliminates the need to represent hosts, gateways, clients and network traffic as distinct entities.

LRU and Exp1 were compared under a number of varying conditions, based upon real traffic examples, but generated randomly from web server statistics. The performance metric used was the perceived retrieval rate (the throughput to the client) which was determined by the number of bytes transferred (or, ideally, retrieved from the cache) divided by the total amount of time for document retrieval. Clearly, if most documents were not downloaded, the total time of transfer would be reduced significantly. Document or byte hit rates might have provided similar results, but these are qualitative metrics and can be misleading.

Under light loads, when there was space for documents to be cached, there was little or no difference between the two techniques; it was expected that during off-peak periods, LRU

might perform very slightly better than Exp1, due to the need for Exp1 to maintain document life histories. However, the difference in overheads were negligible, as network costs were the most significant factor, being several orders of magnitude higher than the extra computation and storage retrieval costs<sup>7</sup>. Significant differences between the two techniques were only apparent when the decision was made of which files to keep or remove from the cache. Once the cache became 'saturated', the adaptive approach soon out-performed LRU.

Actually, the algorithm tested was less than optimal, because it only used the life history information, calculated from the previous transactions, rather than including the current recency information as well. While this allowed a one-time calculation of document life histories to simplify the prioritisation of cache entries in order of importance, it could allow a recently popular document to be stuck in the cache indefinitely, if it was no longer requested. An experiment was designed to exacerbate the situation in WebAgent: Document usage underwent a periodic, catastrophic change in document usage. The recently most popular files suddenly becoming the least frequently requested. Although Exp1 showed a reduction in perceived retrieval rate, LRU had a corresponding (but larger) fall. Hence, the adaptive technique still out-performed the more traditional approach, but this is not necessarily conclusive. It is possible in chaotic conditions that both techniques might converge in performance; any slight advantages of the adaptive approach being cancelled out with the extra overhead involved.

A more optimal approach would involve a hybridisation of recency and frequency, which could be determined by the same exponential smoothing technique – the time since the last request would be recalculated with the pre-determined mean time to next request giving a

---

<sup>7</sup> By comparison, Squid, one of the most popular cache programs, will attempt to clear stale documents by randomly picking a subset of all the cached documents, and deleting the oldest, rather than keeping a complete list of all the cached documents, as this would cause unacceptable 'lag'. Under heavy loading, this is barely able to keep up with the pace of web transactions.



weighting that would gradually increase the likelihood of the file being discarded. Only when a new request for the document came in would the history be modified to prevent false creeping up of the usage information every time the cache was reordered. However, this would also increase the computation required to determine which files to remove.

This section has dealt specifically with the structure of WebAgent, as it is only mentioned in passing in many of the papers included in the portfolio. In the following section, more details are given of the exponential smoothing technique utilised by Exp1, and the results of using WebAgent to perform experimental evaluation of Exp1 in comparison with LRU.

### **3.4 Description of the WebAgent papers**

The following papers describe the chronological development of Exp1. The first paper describes Exp1 from its initial conception as an algorithmic technique, to its implementation within the WebAgent Testbed. An empirical evaluation of its performance is made in the second paper. Finally, the last two papers describe the mathematical underpinning to Exp1, and attempt to explain its performance improvements over LRU.

#### **3.4.1 Paper 9: “Intelligent Control of Dynamic Caching Strategies for Web Servers and Clients”**

**Abstract:** Web pages are cached to reduce network load; various strategies have been adopted which are centred around hierarchies of proxy servers. However, this approach introduces coherence problems. If possible, documents should be kept ‘coherent’ to prevent delivery of out of date, or ‘stale’ pages. We suggest that current proxy server and client caching techniques are inadequate for future exponential growth of the Internet, as they do not attempt to address the dynamics of document selection and modification. We propose an intelligent dynamic caching technique to model document life histories. This work addresses the coherence problem with particular emphasis on strategies suitable for client browser cache management.

#### **Conclusions of the paper**

In this paper, various techniques for management of client and server based web caches were examined, and the case made for an intelligent agent that models the usefulness of web objects by evaluation of document life histories. This agent, while still less than optimal,

shows an improvement in the handling of web objects over existing techniques, such as LRU. The WebAgent simulation has reproduced results that suggest that the frequency of requests for a document, rather than file size, is more relevant to the management of web caches, and that even rough estimates of document request rates can significantly improve performance. Furthermore, the dynamic nature of the proposed approach, provides ever-improving performance, by adapting itself to frequently variable network use and performance. Although these techniques have been aimed primarily at client caches, the authors believe that they are appropriate and scalable to proxy servers.

### **Commentary**

The main emphasis for this paper was to outline the mathematical underpinning of the proposed cache management algorithm implemented in Exp1, and to propose the concept of document life histories. Experimental results were presented, but no quantitative assessment was made of how or why this technique performed better than the traditional approach. Little mention is made of the structure of WebAgent, except in the final conclusions, as the audience were primarily internet, rather than agent researchers.

### **3.4.2 Paper 10: “An Adaptive Mechanism for Web Browser Cache Management”**

**Abstract:** Current proxy server and client caching techniques do not incorporate the dynamics of document selection and modification. The adaptive model that is proposed in this article uses document life histories to optimise cache performance.

### **Conclusions of the paper**

Modelling Web object usefulness based on estimates of document request rates shows 5-10% improvement in performance over existing cache management techniques such as LRU. Furthermore, this model is dynamic and should improve performance continuously as it adapts to variable network use and performance. Although this technique has been aimed primarily at web browsers, we believe it may also be appropriate for proxy servers. We used the WebAgent simulator to obtain our results. WebAgent provides an environment for simulating file requests (modelling trends in user interest in documents), download delays (seasonal and catastrophic changes in network performance), and document modifications.

Several improvements are planned for the WebAgent simulator that will increase its accuracy. These include better usage modelling and more accurate simulations of network performance.

Future work with the WebAgent simulator includes the evaluation of pre-fetching algorithms, which can use “off-peak” times to maintain cache coherence for frequently used documents. We also plan to develop the simulated results into real client and server-based proxy caching agents, which can analyse geographical trends in user access to documents and improve performance to distributed mirror sites.

### **Commentary**

This treatment was empirical, but the use of an agent-based approach allowed for in-depth analysis by allowing the network behaviour to be monitored and modified. Originally, the paper was submitted to the journal for a special issue on ‘internet’ agents, but was accepted for a subsequent general issue. Therefore, the structure was modified to emphasise the new caching technique and its explanation, with less mention being made of the WebAgent Testbed. Publication led to several requests by the editors to peer review other adaptive caching papers for the journal.

### **3.4.3 Paper 11: “Intelligent adaptive web caching using document life histories: A comparison with existing management techniques”**

**Commentary:** These are the slides used to present the work to the conference. An extended version of the presented paper, which was modified in light of comments by several of the delegates, is provided in the Section 3.4.4. The author received praise from some quarters, especially for making the underlying strategies of existing caching algorithms accessible, as they are often steeped in mathematical expressions. However, the author also received pointers for material that had been overlooked. Due to the nature of the conference, some description of the simulation itself would have been desirable, as the technique was a novel one, but discussion was limited. However, a number of commercial representatives were interested in reproducing the reported findings, with the hope of future inclusion in their products; see accompanying correspondence.

#### **3.4.4 Paper 12: “Exp1: a comparison between a simple adaptive caching agent using document life histories and existing techniques”**

**Abstract:** Hierarchical storage of web pages in proxy server and client browser caches

introduce coherence problems, which require cache management techniques which are both accurate and computationally efficient. We suggest that current approaches, such as the most common Least Recently Used (LRU) technique, are inadequate for future network loads, as they do not incorporate the dynamics of document selection and modification. We propose the use of an intelligent, adaptive cache management technique to overcome the coherence problem by using document life histories to optimise cache performance. This work addresses the use of damped exponential smoothing to model accurately the frequency of file requests and modifications, in order to predict the future value of cached files. Finally, we make a mathematical analysis of LRU in comparison with our technique, showing how and why the use of document life histories is a more effective cache management technique without imposing major computational overheads.

#### **Conclusions of the paper**

In this paper, the case has been made for intelligent modelling of the usefulness of web objects by evaluation of document life histories. We propose such a system, which shows an improvement in the handling of web objects over existing techniques, such as LRU. Estimates of document request rates, can significantly improve cache performance. Furthermore, the dynamic nature of our approach, should provide ever-improving performance, and a system which can adapt itself to variable network performance. Finally, we have examined the mathematical underpinnings of both LRU and Exp1, and determined that while the former is computationally inexpensive, it is not the optimal solution for intelligent, adaptive cache management.

Future work will include an investigation into the dynamic calculation of  $\alpha$  to optimise cache performance. It is also possible that exponential smoothing might be applied to predicting file modification times to estimate document ‘shelf life’. Further improvements have been identified for the WebAgent simulator, which will serve to improve the accuracy of the

simulation. Finally, an intelligent dynamic caching agent should be created for an existing web browser, to evaluate performance of an actual user on a real network.

### **Commentary**

This paper is an extended version of the paper, entitled “Intelligent web caching using document life histories: A comparison with existing cache management techniques,” presented to the 3<sup>rd</sup> International Caching Workshop (Manchester, June 1998). Some correspondence was generated from this publication, including a request for access to WebAgent to test the algorithm proposed in a recent PhD thesis; see accompanying correspondence.

### **3.5 Conclusions and Future Work**

In WebAgent, the continual on-line, real-time construction of the document life histories is the emergent result of implicit learning within Exp1. Agent communication with the local user environment and with other caches, is matching the distributed nature of the problem implicitly. Therefore, the validity of the caching strategy is a function of the information achieved through repeated document transactions.

With Exp1, data mining and cache management is occurring concurrently in a non-static environment. Information ‘mined’ at each cache is more complete and prioritised in a justifiable way. The exponential smoothing technique is more robust than LRU for incorrect estimates created by sudden changes in document usage; effectively making use of cumulative reinforcement. It is difficult to have an adaptive strategy that does not oscillate. LRU adopts a “memoryless strategy”, which is adaptive but oversensitive to change. Exp1 manages to achieve the former, without falling foul of the latter.

The technique estimates smoothed averages in an asynchronous way, but builds and uses more information than its competitors. The frequency of updates is based solely upon actual transactions, which is better than regular update requests or validity checks that have an arbitrary period, and may increase traffic unnecessarily. The knowledge gathered can be

exploited for a variety of different purposes: sorting intelligent histories for browsers, making informed choices for acceptable staleness, and determining the cache threshold level.

There is a common misconception in the literature, that software agents must be deliberative, while reactive agents may only be mobile robots; features that are commonly assumed to be pre-requisites for systems to be considered 'real' agents. The term 'agent' is justified in the context of Exp1, because it behaves 'reactively' in choosing from a pre-defined set of responses, even though the agency within the cache is not mobile. Furthermore, the behaviour is also partially 'deliberative', because Exp1 maintains an internal representation of its environment, based upon probabilities derived by observation of local traffic statistics. Technically, therefore, Exp1 should be classified as a 'hybrid' architecture.

### **3.5.1 Evaluation of Exp1**

Is Exp1 an agent? Foner (1993) identifies the following as being required for a system to be considered a 'true' agent: Autonomy; Personalisability; Discourse; Risk & Trust; Domain; Graceful degradation; Co-operation; Anthropomorphism; and Expectations. Although his view of agency is primarily aimed at interface agents, hence the need for discourse and anthropomorphism, the other factors are also applicable for sole agents, as well as multi-agent systems, and for reactive as well as deliberative agents.

Exp1 clearly has the capacity for autonomy<sup>8</sup>. It is personalisable in that it will adapt to the documents which are being requested, by using rudimentary reinforcement learning; for caches situated in a web browser, this is more extreme, in that the cache represents the individual user preferences. Exp1 does not, at present, exhibit much discourse with the user. However, this is a possibility for the future as there are situations where, for the end user at least, a dialogue could exist between the user and the cache: to prioritise performance against

---

<sup>8</sup> Exp1 maintains a measure of document usage, even if the document is not cached, has been forced from the cache, or is never cached due to its size. The threshold could be set dynamically, allowing for temporary increases, if that would improve cache performance; e.g. the Netscape scenario described by Bennett (1998).

staleness, for example, or to prompt a client that a document is likely to have changed, etc. The dialogue for a proxy server cache could be to set acceptable staleness, or for determining the level of dynamic pre-fetching, etc. Clearly, this should be explored in future work.

The risk & trust in an agent is determined upon what task has been delegated to the agent. In the case of a risk assessment of Exp1, this is fairly straightforward; caching is not a critical activity, and people do not give it a second thought. However, the delivery of stale documents without a corresponding warning, could undermine trust in a proxy server. Exp1 makes an estimate of the mean time to the next document change, which gives at least some information to determine how likely such a change could be. An alternative would be to serve the cached document, then inform the user about the newer version, if one is available once the validation had been received.

The possibility of preserving life histories while a cache is off-line, might be a useful thing in itself, as it can take weeks to 'build' a cache back up. With a record of the most popular documents, Exp1 could set about using free bandwidth to reinstate the most important files. While this might involve the transfer of a document that will not be subsequently requested by a client, it would reduce downloads overall, by choosing download times that corresponded to light network loads for the original server (e.g. while the USA is asleep).

The domain of Exp1 is merely that of caching, which is a partially visible problem, in that only web transactions can provide information, there being no other communication that would not needlessly increase bandwidth. Peering a cache with its neighbours, allows the possibility for collaboration between multiple copies of Exp1. It could be envisaged that an Exp1 agent could swap document life histories with peers, although it is not clear what advantage this might have for cache performance. This has yet to be explored. Otherwise, the problem is conceptually simple, without the need for the agent to make accurate assessment of other complex entities.

This approach also allows for graceful degradation, because the worst performance for Exp1 is equivalent to LRU, the currently most popular cache management technique. Features of Exp1 might actually improve upon LRU, which is adaptive but unintelligent and unable to provide useful information to users. Co-operation between internet users and providers is implicit as network considerations are uppermost in the use of a cache; saving network resources, at the expense of hardware and storage costs. Co-operation between cache and client can be more explicit, when a dialogue is possible. Again, the current state of Exp1 means that this possibility is yet to be explored. However, a similar commercial system, DynaCache, has shown that dynamic recognition of intelligent caches can lead to active collaborative actions [INFOLIBRIA (1998)].

The expectation of Exp1 is, for end users, less delays in document retrieval, and for proxy servers the reduction in bandwidth and the effective use of storage media. In this respect, there is clearly a useful task to perform in the background. However, there is the possibility for more informed decisions to be made by the user, which could potentially alter their use of the Internet as a resource<sup>9</sup>.

Letizia is a tool for aiding web browser navigation that, although primarily a search engine, “tracks the user's browsing behaviour... and tries to anticipate what items may be of interest to the user” [Lieberman (1995)]. Letizia uses simple heuristics and past behaviour to approximate user interest, and then can present recommendations for further study. While this task is different to that of caching, there are a number of similarities. While the scale is larger, and the emphasis is upon document requests rather than contents, Letizia and Exp1 are still attempting to model document usage based upon previous transactions. In both cases, the value of a document attenuates over time.

---

<sup>9</sup> The use of a video recorder has revolutionised TV usage, much more than Satellite or Cable. Time independence and convenience is much more useful than freedom of choice.



This raises the question of how to implement the Exp1 algorithm inside a real cache management system. It is suggested that files might be ordered by the frequency of access, so that the value of the historical usage can be included in the assessment of which file to remove. The first time a file is cached, it is worth giving a certain amount of time before removing it, as if it is immediately discarded, this could cause inefficiencies as a popular document might be removed in error. Furthermore, it is a common problem in caching to identify documents that are used only once, but haunt the cache until they 'time out.' This can often be a crucial factor in cache maintenance. Finally, once-only documents should not be put in the disk cache [Kurcewicz (1998)]. If they are the oldest in memory, they should be discarded. Therefore, new entries could be placed initially at the head of the queue. Once-only files, having a history of zero, would be more likely to be discarded over time, but initially would be evaluated as frequently used documents. However, existing entries, instead of being brought to the head of the list (as effectively happens with LRU), could be swapped with the nearest neighbour. This would lead to the order of the cache implicitly reflecting the frequency of use of the entries.

It might be possible to predict the current performance of the network between the cache and the source, by evaluating the response time for validation packets, and building a model of typical and best network/server performance in the day for daily updates, in the week for weekly updates, etc. However, the chaotic nature of the internet might make such predictions misleading<sup>10</sup>. Nevertheless, performance improvements might be possible, utilising document life histories to determine whether a file should be stored in memory or on disk, or whether to check a file for validity, given the mean time between document modifications.

It might also be possible to use a 'global' request rate for a cache, to allow prediction of current server load. In times of saturation, it might be possible to forego validation requests

---

<sup>10</sup> At this level it is difficult to know whether a delay is due to a heavily loaded path, a broken gateway node, or just the load on the server at the time of the request being sent and received.

and just serve up the cached version of a document. Similarly, if the validation request took more than a certain period of time the cached version could be used, with a warning; an equivalent of this occurs in browsers when the host server is unobtainable.

### **3.5.2 Evaluation of WebAgent**

Reddy & Moon (1995) define several criteria for evaluating ABSEs [ibid pp395-6]:

Appropriateness and timeliness of solutions; handling unplanned events and incomplete knowledge; independent representation of sensors and effectors; effective modelling of time within the simulation; and, graceful performance under concurrency.

**Appropriateness and Timeliness of Solutions** - The purpose of WebAgent is to simulate typical transactions between information producers and consumers on the Internet. This allows researchers to compare various caching strategies, under repeatable, controllable experimental conditions. Therefore, appropriateness and timeliness are evaluated by direct comparison between different caching algorithms.

**Handling Unplanned Events and Incomplete Knowledge** - Since WebAgent is simulating Internet transactions that are hard to predict, and prone to radical changes, the system agents (the Request Generator and Client Agents, the Update Generator and server Agents, and the Network Agent) must actually generate unplanned (and unplannable?) events. Furthermore, Cache agents within WebAgent must work with incomplete knowledge, as real world caches only have access to limited information.

**Independent Representation of Sensors and Effectors** - Cache Agents must only communicate via reasonable means, given the limitations of their real world counterparts, which only 'sense' the world via document requests from clients, and the responses from servers. Therefore, a secondary function of the system agents is to shield details of the simulation from the various caching systems.

**Effective Modelling of Time within the Simulation** - The representation of time in WebAgent is discrete, rather than continuous. However, given the domain is based upon discrete events (such as document requests arriving at a server), this is adequate provided that the time increments are sufficiently small.

**Graceful Performance under Concurrency** - WebAgent does not explicitly use concurrency for its system agents, as it needs to allow for comparison between different caching strategies. Cache agents are effectively run in parallel, though this is merely simulated pseudo-concurrency, so that they all appear to execute on a 'level playing field'.

Further development of WebAgent will address fixed bandwidth data acknowledgement and retransmission errors to allow future simulations to consider quality of service issues. The possibility of implementing packet level transmission, and standard network routing techniques, such as IP masking, will allow low-level experimentation into more general network performance techniques, such as document migration.

## **4.0 Commentary**

### **4.1 Link between the two projects**

RAPIDO is a toolkit for deliberative agents, and stands as a finished piece of work in its own right which, though dated by today's standards, still has many features that are currently being developed within the field of MAS architectures and toolkits. WebAgent, on the other hand, is an agent-based simulation testbed for evaluating different cache management techniques. It represents work-in-progress, with a number of unanswered questions, and promising future directions of research. Both projects are linked by the consideration of low level communications and the implications for building agent applications. The first is a real implementation of parallel-distributed processes passing messages. The second is a conceptual representation of such a schema, for the purposes of evaluating performance.

The WebAgent simulator, re-creates the environment of a network of users and service providers. This work makes use of agents to help problem solving, rather than considering the construction of agent systems. Furthermore, it provides a methodology for evaluating the results of cache activity, and making comparisons with between different cache management techniques. Finally, the Exp1 caching agent clearly demonstrates rudimentary learning of stimulus/reaction patterns, based upon past behaviour, which allows instant, pre-determined, but adaptable decisions.

In that respect, the issue of agency becomes implicit and secondary to that of seeking solutions to novel and traditionally difficult to solve problems. The need for an agent-based approach to this simulation, was that of expediency; it is time-consuming, and unnecessary to simulate packets, nodes and bandwidth to evaluate caching strategies. However, both the conceptual network model and the implementation of WebAgent, were inspired and driven by experience gained from developing and using RAPIDO. The construction of the two projects, therefore, show the development across the gamut of agent types: from deliberative to reactive, symbolic to situated, experimental testbed to real world application.

The next section discusses the original contribution that has been made by investigating intelligent agents and agent technologies.

## **4.2 Contribution to Knowledge**

While the two projects have broken new ground in different ways, the principle contributions to the agents field are inter-related, and have been threefold:

- 1) Cross-pollination from related disciplines has much to offer by preventing 're-invention of the wheel.'

RAPIDO and WebAgent have both benefitted from the fields of Distributed, Concurrent and Real-Time Systems, and Mathematics, respectively. Furthermore, the development of Exp1 has also provided evidence that simple agents can be effective tools for improving performance of a distributed network, thus showing that Agent Technologies can return the favour by being of use in other fields.

- 2) Abstraction in the representation of agents as communicating processes has provided both a generic life-cycle model for the acquisition and design of MAS applications, and evidence that agents can simulate complex real-world problems in an elegant manner.

The generic approach to the specification of agent applications, in RAPIDO, and object-oriented rapid prototyping without recourse to low-level programming, allow a greater degree of freedom of choice over the implementation paradigm. Further investigation into the development of WebAgent has introduced a new agent-based methodology for investigating performance of internet traffic, without recourse to trace-based, or packet-level simulations.

- 3) Use of Agent-Based Simulation Environments (ABSEs) provides an effective tool for evaluating complex problems. Comparison of the performance of different agent architectures may only be possible when experiments are controlled and repeatable [Hanks et al (1993)].

The ability of RAPIDO to evaluate the relative performance of each paradigm for a specific domain has been tested by the experience of rapid prototyping a working MAS application, AMNESIA. These techniques are of use to anyone hoping to construct optimised MAS, for specific problem areas. The potential for using ABSEs has been proven to be effective, with the discovery that an adaptive agent can outperform existing cache management techniques.

WebAgent shows significant improvement over previous techniques used to assess the performance of new caching algorithms by being flexible, adjustable and repeatable.

The knowledge gained during these projects has been disseminated by the author in a number of ways. The papers included as part of the accompanying portfolio have allowed peer review. These publications, in addition to this overview, reflect the improvement in understanding that subsequent discussion within the academic community and feedback from colleagues have produced, showing the development of ideas from their earliest origins, through exploration and experimentation, to the final analysis and dissemination.

RAPIDO has attempted to fill a gap in the development of DAI and MAS applications, requiring the author to have an in-depth knowledge of real-time, concurrent and distributed systems techniques for inter-process communications, control and synchronisation, and to apply this to the low-level implementation of CoCo-POP and RAPIDO. The technical insights gained from this research were invaluable in shaping the development of the network simulation within WebAgent, and inspiring the development of the Exp1 technique for cache management. However, it is the author's belief that this will also serve to inspire the future investigation of Agent Technologies.

### **4.3 Future Work**

The advent of the Internet presents problems that may only be addressed by agent-based techniques. It is generally recognised that optimising network performance is a time varying, distributed, resource constrained problem; further complicated for the Internet by competing, incompatible user requirements. Reactive agents hold much promise in this new and growing field, as they are adaptive and better placed to provide timely solutions in constrained circumstances. An important aim is to develop Exp1 into a caching agent application for an existing browser (such as Netscape) to explore:

- 1) prediction of document popularity to allow intelligent sorting of browser bookmarks and/or histories, or dynamic pre-fetching for frequently used documents;

- 2) prediction of likely document changes to allow cached documents to be served with a 'health warning' depending upon the level of acceptable staleness;

Ultimately, this will culminate in the construction of an intelligent dynamic agent for a widely used proxy cache (such as SQUID), to evaluate:

- 1) server-based analysis of geographical trends in user access to popular documents, to improve pre-emptive distribution of documents to mirror servers – document migration – and subsequent maintenance of propagating correct versions after modification;
- 2) prediction of network performance to allow pre-fetching algorithms to make use of 'off peak' times to maintain cache coherency .

A number of future improvements have been identified for the WebAgent simulation; including extending the simulation to include multiple, geographically distributed servers, with more realistic weightings for network performance, and standard network routing techniques, such as IP masking. This will allow further exploration of the application of simple adaptive agents to the regulation and management of networks to consider quality of service issues. The application of WebAgent, a simulation of the Internet has proven the effectiveness of using agent-based tools for performing evaluation experiments. However, it is hoped that this will contribute to the development of MAST (Multi-Agent Simulation Testbed), a simulation testbed for other situated agents, such as mobile robots.

Historically, ABSEs have been used only for investigating the validity of applying agents to a problem, rather than as tools in their own right to solve real world problems. Currently, research into the application of agents to this important area have been based upon planning and BDI architectures [Howe (1994); Georgeff & Rao (1998)]. One such area is safety-critical systems [Storey (1996)]. It is the author's belief that there is a case for examining the application of 'weak agents' and agent-based simulation to safety-critical systems. This requires further investigation into a generic model of agents, based upon communication requirements and object-oriented design techniques. It is hoped that this model may lead to the proposal of a new lifecycle for the development of Agent Technologies.

## **5.0 References**

- Almeida, V. & de Oliveira, A. (1996)**, "On the fractal nature of WWW and its applications to cache modelling," Technical Report TR-96-004, Computer Science Department, Boston University, 111 Cummington Street, Boston, MA 02215.
- Allen, J.F. & Perrault, C.R. (1980)**, "Analyzing intention in utterances," in 'Artificial Intelligence,' Vol. 15, No. 3, pp143-178.
- Arlitt, M.F. & Williamson, C.L. (1996)**, "Web server workload characterisation: The search for invariants," ACM SIGMETRICS Conference, 1996.
- Appelt, D.E. (1985)**, "Planning English Sentences," Cambridge University Press, New York.
- Bennett, S. (1998)**, Lancaster University, Personal communication.
- Bond, A.H. & Gasser, L. (1988)**, (eds), "Readings in Distributed Artificial Intelligence," Morgan Kaufmann.
- Brooks, R. & Connell, J.H. (1986)**, "Asynchronous distributed control system for a mobile robot," SPIE 727.
- Cohen, P.R. & Perrault, C.R. (1979)**, "Elements of a plan-based theory of Speech Acts," in 'Cognitive Science,' Vol. 3, No. 3, pp177-212.
- Connah, D. (1991)**, "Why we need a new approach to the design of agents," AISBQ 76.
- Crovella, M.E. & Bestavros, A. (1995)**, "Explaining world wide web traffic self-similarity," Technical Report TR-95-015, Computer Science Department, Boston University, 111 Cummington Street, Boston, MA 02215.
- Cunha, C.R., Bestavros, A. & Crovella, M.E. (1995)**, "Characteristics of WWW client-based traces," Technical Report BU-CS-95-010, Computer Science Department, University of Boston, 111 Cummington Street, Boston, MA 02215.
- Davis, R & Smith, R.G. (1983)**, "Negotiation as a Metaphor for Distributed Problem Solving," Artificial Intelligence 20, P63-109.
- Demazeau, Y. & Müller, J-P. (1991)**, "From reactive to intentional agents," in Demazeau, Y. & Müller, J-P. (eds), 'Decentralized Artificial Intelligence 2,' Elsevier, pp3-14.
- Di Caro, G. & Dorigo M. (1998)**, "AntNet: Distributed Stigmergetic Control for



Communications Networks,” in ‘Journal of Artificial Intelligence Research (JAIR),’ Vol. 9, pp317-365.

**Dijkstra, E.W. (1975),** “Guarded Commands, Non-Determinism and Formal Derivation of Programs,” in ‘Communications of the ACM,’ 18, 8, Aug 1975, pp453-457.

**Durfee, E., Lesser, V.R. & Corkill, D.D. (1987),** “Co-operation through Communication in a Distributed Problem Solving Network” in Huhns, M.N. (ed), ‘Distributed Artificial Intelligence’ Pitman pp29-58, 1987.

**Englemore R.S. & Morgan A.J. (1988),** “Blackboard Systems,” Addison-Wesley, London.

**Ferber, J. (1989),** “Eco problem solving: How to solve a problem by interactions,” in ‘Procs of the 9<sup>th</sup> International Workshop on Distributed Artificial Intelligence,’ Seattle.

**Ferber, J. (1996),** “Reactive distributed artificial intelligence: Principles and applications,” in [O’Hare & Jennings (1996)], p287-314.

**Fisher M. & Wooldridge M.J. (1993),** “Specifying and verifying distributed intelligent systems,” in Filgueiras, M. and Damas, L. (eds.), ‘Progress in Artificial Intelligence - Proceedings of the 6<sup>th</sup> Portuguese Conference on AI,’ Portugal, October 1993, (LNAI Volume 727), Springer-Verlag.

**Foner, L.N. (1993),** “What’s an agent, anyway? A sociological case study,” Agents Memo 93-01, Agents Group, MIT Media Lab, 20 Ames Street, Cambridge, MA 02139.

**Genesereth M.R., Ginsberg, M.L. & Rosenschein, J.L. (1984),** “Co-operation without Communication,” Report HPP-84-36 September 1984, Stanford Heuristic Programming Project, Stanford University.

**Georgeff, M. & Rao, A. (1998),** “Rational Software Agents: From Theory to Practice,” in [Jennings & Wooldridge (1998)], pp139-160.

**Hanks, S., Pollack, M. & Cohen, P.R. (1993),** “Benchmarks, Test beds Controlled Experimentation and the Design of Agent architectures,” in ‘AI Magazine,’ Vol. 14, No. 4, Winter, pp17-42.

**Hansen, P. Brinch (1973),** “Operating System Principles,” Prentice Hall, New Jersey.

- Hansen, P. Brinch (1978)**, "Distributed Processes : A Concurrent Concept," in 'Communications of the ACM', 21,2 pp934-941.
- Hern, L.E.C. (1987)**, "On Distributed Artificial Intelligence," in 'The Knowledge Engineering Review,' Vol. 3, No. 1, Mar. 1988, pp. 21-57.
- Hewitt M. (1977)**, "Viewing Control Structures as Patterns of Passing Messages," in 'Artificial Intelligence' , Vol. 8, No.3, pp323-364.
- Hewitt, C. & Liebermann, H. (1984)**, "Design issues in parallel architectures for Artificial Intelligence," Proceedings of the 28th IEEE Computer Society International Conference, San Francisco, CA. pp418-423.
- Hoare, C.A.R. (1974)**, "Monitors : An Operating System Structuring Concept," in 'Communications of the ACM' 17,10 pp549-557.
- Hoare, C.A.R. (1978)**, "Communicating Sequential Processes," in 'Communications of the ACM,' 21, 8, pp666-677.
- Howe A.E. (1994)**, "Improving the Reliability of Artificial Intelligence Planning Systems by Analyzing their Failure Recovery," Computer Science Department, Colorado State University, Fort Collins, CO 80523.
- Howe A.E., Hart, D.M. & Cohen P.R. (1990)**, "Addressing Real-Time Constraints in the Design of Autonomous Agents," in 'The Journal of Real-Time Systems,' Vol. 1, pp81-97.
- Huhns, M.N. & Singh, M.P. (1998)**, (eds.), "Readings in Agents," Morgan Kaufmann.
- Ichbiah, J.D. (1980)**, "Reference Manual for the ADA Programming Language," United States Dept. of Defence.
- INFOLIBRIA (1998)**, "DynaCache Distributed network caching technology," InfoLibria White Paper, in 'Proceedings of the 3rd International Workshop on WWW Caching,' June 15-17th, Manchester, UK.
- INSIGHT (1986)**, "The INSIGHT Blackboard Experiment Information Pack," Systems Designers Ltd.
- Jennings, N.R. (1994)**, "Co-operation in Industrial Multi-Agent Systems," World Scientific Publishing, London

- Jennings, N.R. & Wooldridge, M.J. (1998)**, (eds.), "Agent Technology : Foundations, Applications, and Markets," Springer Verlag.
- Kaelbling, L.P., Littman, M.L. & Moore, A.W. (1996)**, "Reinforcement learning: A survey," in 'Journal of Artificial Intelligence Research,' Vol. 4, pp237-285.
- Kurcewicz, M., Sylwestrzak, W. & Wierzbicki, A. (1998)**, "A Filtering Algorithm for Proxy Caches," 'Proceedings of the 3rd International Workshop on WWW Caching,' June 15-17th, Manchester, UK.
- Lenat D.B. (1975)**, "BEINGS : Knowledge as Interacting Experts," in 'Proceedings of the 1975 International Joint Conference on A.I.,' pp126-133.
- Lesser V.R., Fennell, R.D., Erman, L.D. & Reddy, D.R. (1975)**, "Organisation of the HEARSAY II Speech Understanding system," in 'IEEE Transactions on Acoustics, Speech and Signal Processing,' Vol.ASSP-23 No.1, Feb., pp11-24.
- Lesser V.R. & Corkill D.D. (1981)**, "Functionally Accurate / Co-operative Systems" in 'IEEE Transactions on Systems, Man and Cybernetics,' Vol. SMC-11 No.1 pp81-96.
- Lieberman, H. (1995)**, "Letizia: an agent that assists web browsing," in 'Procs of the International Joint Conference on Artificial Intelligence (IJCAI '95),' Montreal, pp924-929.
- Martin J. (1981)**, "Computer Networks and Distributed Processing," Savant, Carnforth.
- Moon, J. (1997)**, "An investigation into the use of Multi-Agent Systems in marine simulator instructor stations," PhD Thesis, May 1997, School of Computing, University of Glamorgan, Trefforest, Pontypridd, Mid Glamorgan, CF37 1DL, Wales, UK.
- Moy, J.T. (1998)**, "OSPF Anatomy of an Internet Routing Protocol," Addison-Wesley.
- Muller, J.P., Wooldridge, M.J. & Jennings, N.R. (1997)**, (eds), "Intelligent Agents III : Agent Theories, Architectures, and Languages : Ecai'96 Workshop," Procs of ATAL, Budapest, Hungary, Springer Verlag.
- O'Hare, G.M.P. (1987)**, "New Directions in Distributed Artificial Intelligence," 2nd International Expert Systems Conference, London, Learned Information (Europe) Ltd.
- O'Hare, G.M.P. & Jennings, N.R. (1996)**, (eds) "Foundations of Distributed Artificial Intelligence," Sixth-Generation Computer Technology Series, Wiley.

- Pebody, M. (1993)**, "How do you choose your agents? How do you distribute your processes?," in Steels L. (ed) 'The biology and technology of autonomous agents' Springer-Verlag pp345-376.
- Perkins, K.A. & Machin, C.H.C. (1997)**, "Entity scheduling for distributed systems," Loughborough University,  
<<http://www-staff.lboro.ac.uk/~cokap1/Entity Scheduling/Entity Scheduling.html>>
- Pressman R.S. (1982)**, "Software Engineering: A Practitioner's Approach," McGraw-Hill
- Reddy D.R., Erman, L.D., Fennell, R.D. & Neely, R.B. (1973)**, "The HEARSAY Speech Understanding System : An Example of the Recognition Process," in 'Procs of the 3rd Int. Joint Conference on A.I.,' pp185-193.
- Rosenschein J.S. & Genesereth M.R. (1984)**, "Communication and Co-operation," Report HPP-84-5, Stanford Heuristic Programming Project, Stanford University.
- Rosenschein, J.S., Ginsburg, M. & Genesereth, M.R. (1986)**, "Cooperation without communication," in 'Proceedings of 1986 Conference of the American Association for Artificial Intelligence,' pp51-57.
- Searle, J.R. (1969)**, "Speech Acts: an essay in the philosophy of language," Cambridge University Press.
- Singh, S.P., Jaakkola, T. & Jordan, M. (1994)**, "Learning without state estimation in partially observable Markovian decision processes," in 'Proceedings of the Eleventh Machine Learning Conference,' Morgan Kaufmann, pp284-292.
- Sloman, A. & Poli, R. (1996)**, "SIM\_AGENT: A toolkit for exploring agent designs," in 'Proceedings of IJCAI'95 Workshop on Agents, Theories, Architectures and Languages (ATAL'95),' Springer-Verlag, Lecture Notes in Computer Science,
- Smith R.G. (1978)**, "A Framework for Problem solving in a Distributed Processing Environment," Report HPP-78-28 December 1978, Stanford Heuristic Programming Project, Stanford University.
- Smith R.G. & Davis R. (1981)**, "Frameworks for Co-operative Problem Solving," in 'IEEE Trans. on Systems, Man and Cybernetics,' Vol. SMC-11 No.1 January 1981.

- Steeb, R., Cammarata, S., Hayes-Roth, F.A., Thorndyke, P.W. & Wesson, R.B. (1981),** “Architectures for distributed intelligence for air fleet control,” Technical Report R-2728-ARPA, Rand Corporation, Santa Monica.
- Storey, N. (1996),** “Safety-Critical Computer Systems,” Addison-Wesley.
- Williams, S., Abrams, M., Standridge, C.R., Abdulla, G. & Fox, E.A. (1996),** “Removal policies in network caches for WWW documents,” Proceedings of the ACM SigComm Conference, August 1996, Stanford.
- Wooldridge, M. & Jennings, N.R. (1994),** (eds.), “Agent Theories, Architectures and Languages: a survey,” Proceedings of the 1994 Workshop on Agent Theories, Architectures and Languages, ECAI’94, Amsterdam, August, pp1-32.
- Yang, J.D., Huhns, M.N. & Stephens, L.M. (1985),** “An Architecture for Control and Communications in Distributed Artificial Intelligent Systems,” in ‘IEEE Trans. on Systems, Man and Cybernetics,’ Vol.SMC-15, No.3 May, pp316-326.
- Young, S.J. (1982),** “Real-Time Languages,” Ellis Horewood, Chichester.
- Zipf, G.K. (1949),** “Human behaviour and the principle of least-effort,” Addison Wesley, Cambridge.

# **An investigation into the use, application and evaluation of intelligent agents**

***Mike Reddy***

**Appendix: PhD Portfolio**

***May 1999***

### ***Collected Bibliography for Papers in the Portfolio***

- Abrams, M. (1995)**, "Caching Proxies: Limitations and Potentials," in 'Proceedings of the Fourth World Wide Web Conference,' Elsevier, December, pp119-133.
- Agha, G. (1986)**, "ACTORS: A model of concurrent computation in distributed systems," MIT Press.
- Allen, J.F. & Perrault, C.R. (1980)**, "Analyzing intention in utterances," in 'Artificial Intelligence,' Vol. 15, No. 3, pp143-178.
- Avouris, N. (1994) (ed)**, "Distributed Artificial Intelligence: Theory and praxis," Kluwer Academic Press.
- Appelt, D.E. (1985)**, "Planning English Sentences," Cambridge University Press.
- Barrett, R., Ramsay, A. & A. Sloman (1986)**, "POP-11: A practical language for artificial intelligence," Ellis Horewood
- Berg-Cross, (1989)**, "Acquiring and managing knowledge using a conceptual structures approach: Introduction and framework" in 'IEEE Trans. Sys, Cyb Man,' Vol. 19, No. 3, May/June, pp513-527
- Bisiani, R., Alleva, A., Forin, A., Lerner, R. & Bauer, M. (1987)**, "The architecture of the AGORA environment," in Huhns, M.N. (ed) 'distributed Artificial Intelligence,' Pitman, pp99-118.
- Bolot, J-C. & Hoschka, P. (1996)**, "Performance Engineering of the World Wide Web: Application to Dimensioning and Cache Design," in 'Computer Networks and ISDN Systems,' Vol. 28, No. 7-11, pp1397-1405.
- Bond, A.H. & Gasser, L. (1988)**, (eds), "Readings in Distributed Artificial Intelligence," Morgan Kaufmann.
- Bowman, C.M. (1995)**, "The HARVEST Information Discovery and Access System," in 'Computer Networks and ISDN Systems,' Vol. 28, No. 1-2, pp119-25.
- Buchanan, B.G. (1983)**, "Constructing an Expert System," in [Hayes-Roth, Waterman & Lenat (1983)]
- Cammarata, S., McArthur, D. & Steeb, R., (1983)**, "Strategies of cooperation in distributed problem solving," in 'Proceedings of the 1983 International Joint Conference on Artificial Intelligence,' pp767-770.
- Cohen, P.R. & Perrault, C.R. (1979)**, "Elements of a plan-based theory of Speech Acts," in 'Cognitive Science,' Vol. 3, No. 3, pp177-212.
- Cohen, P.R. (1989)**, "Trial by fire: Understanding the design requirements for agents in complex environments," in 'AI Magezine,' Vol. 10, No. 3, pp32-48.
- Corkhill, D.D., (1986)**, "GBB: A Generic Blackboard Development System," in 'Proceedings of the 5th National Conference on AI (AAAI 86), pp1008-14.
- Craig, I.D. (1986)**, "The ARIADNE-1 Blackboard System," in 'The Computer Journal,' Vol. 29, No. 3, pp235-40.
- Cunha, C., Bestavros, A. & Crovella, M.E. (1995)**, "Characteristics of WWW client based traces," Technical Report BU-CS-95-010, Computer Science Department, University of Boston, 111 Cummington Street, Boston, MA 02215.
- Decker, K.S. (1987)**, "Distributed problem-solving techniques: A survey," in 'IEEE Transactions on Systems, Man and Cybernetics,' SMC-17, pp729-740.
- Deen, S.M. (1991) (ed)**, "Cooperating Knowledge-Based Systems," Springer Verlag.
- Dijkstra, E.W. (1975)**, "Guarded Commands, Non-Determinism and Formal Derivation of Programs," in 'Communications of the ACM,' 18, 8, Aug 1975, pp453-457.
- Dingle, A. & Partl, T. (1996)**, "Web Cache Coherence," in 'Computer Networks and ISDN Systems,' Vol. 28, No. 7-11, pp907-20.
- Doran, J. (1990)**, "The MCS multi-agent testbed: developments and experiments," in Deen, S.M. (1991) (ed), 'Cooperating Knowledge-Based Systems 1990,' 3-5 October, University of Keele, UK, Springer Verlag.
- Durfee, E.H., Lesser, V.R. & Corkill, D. (1988)**, "Coherent Cooperation among Communicating Problem Solvers," reprinted in Bond, A.H. & Gasser, L. (1988) (eds) 'Readings in Distributed Artificial Intelligence,' Morgan Kaufmann.
- Englemore R.S. & Morgan A.J. (1988)**, "Blackboard Systems," Addison-Wesley.
- Englemore, R.S. & Nii, H.P. (1977)**, "A Knowledge-Based System for the Interpretation of

Protein X-Ray Crystallographic Data," Report HPP-77-2, Stanford Heuristic Programming Project, Stanford University.

**Erman, L.D. (1981)**, The Design and Example Use of HEARSAY III," in 'Proceedings of the 7th International Joint Conference on AI,' pp409-15.

**Erman, L.D., Lark, J.S. & Hayes-Roth, F. (1988)**, "ABE: An environment for engineering intelligent systems," in 'IEEE Transactions on Software Engineering,' Special issue on AI.

**Ernst, G. & Newell, A. (1969)**, GPS: A Case Study in Generality and Problem Solving," Academic Press, New York.

**Ferber, J. (1992)**, "Using reactive MASs in simulation and problem solving," in [Avouris (1994)].

**Fergusson, I.A. (1994)**, "Integrated Control and Co-ordinated Behaviour: A Case for Agent Models," in [Wooldridge & Jennings (1994)], pp186-99.

**Feigenbaum, E.A. (1977)**, "The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering," in 'Proceedings of the 5th International Joint Conference on AI.'

**Feldmeier, D. (1988)**, "Improving Gateway Performance with a Routing Table Cache," in 'Proceedings of the Conference on Computer Communications (Infocom 88),' IEEE Computer Society Press, March, pp298-307.

**Feyter, A.R. (1990)**, "RTEX: An Industrial Real-Time Expert System Shell," in 'Proceedings of Avignon '90,' Vol. 1, EC2, France.

**Filby, I. (1991) (ed)**, "Proceedings of the SGES Knowledge-Based Systems Methodologies Workshop," London, 3-4th December, SGES Workshop Publishers.

**Foster, C.C. (1982)**, "Real-time Programming – neglected topics," Addison-Wesley.

**Galliers, J. (1988)**, "A Strategic Framework for Multi-Agent Cooperative Dialogue," in 'Proceedings of the 1988 European Conference on AI (ECAI '88),' Pitman.

**Gardner Jr., E.S. (1985)**, "Exponential Smoothing: The State of the Art," in 'The Journal of Forecasting,' Vol.4, No. 4, pp. 1-28.

**Gasser, L., Braganza, C. & Herman, N. (1987)**, "MACE: A flexible testbed for distributed AI research," in [Huhns (1987)] pp119-152.

**Gasser, L. (1989)**, "Implementing Distributed AI Systems using MACE," in [Bond & Gasser (1988)].

**Gevens, A.S. (1983)**, Overview of the Human Brain and Distributed Computing Networks," in 'Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers.'

**Genesereth M.R., Ginsberg, M.L. & Rosenschein, J.S. (1984)**, "Co-operation without Communication," Report HPP-84-36 September 1984, Stanford Heuristic Programming Project, Stanford University.

**Genesereth M.R. & Nilsson N. (1986)**, "Logical Foundations of Artificial Intelligence," Morgan Kaufmann.

**Glassman, S. (1994)**, "A Caching Relay for the World Wide Web," in 'Computer Networks and ISDN Systems,' Vol. 27, No. 2, pp165-73.

**Green, P. (1987)**, "AF: A framework for real time distributed cooperative problem solving," in Huhns, M.N. (ed) 'distributed Artificial Intelligence,' Pitman, pp153-176.

**Hall, L.E., Macauley, L. & O'Hare, G.M.P. (1992)**, "User Role in Problem Solving with Distributed Artificial Intelligent Systems," in Castelfranchi, C. & Werner. E. (1992) (eds), 'Proceedings of MAAMAW '92, the 4th European Workshop on Modelling Autonomous Agents in an Artificial World,' Elsevier.

**Hanks, S. (1993)**, "Benchmarks, testbeds, controlled experimentation and the design of agent architectures," in 'AI Magazine,' Vol. 14, No. 4, pp17-42.

**Hansen, P. Brinch (1978)**, "Distributed Processes : A Concurrent Concept," in 'Communications of the ACM', 21,2, pp934-941.

**Hanson, A.R. & Riseman, E.M. (1978)**, "VISIONS: A Computer System for interpreting Scenes," in Hanson, A.R. & Riseman, E.M.. (1978) (eds), 'Computer Vision Systems,' Academic Press.

**Hayes-Roth, B. (1979)**, "Modelling Planning as an incremental, opportunistic process," in 'Proceedings of the 6th International Joint Conference on AI,' pp375-83.

**Hayes-Roth, B. (1983)**, "The Blackboard Architecture: A General Framework for Problem Solving?" HPP-83-30, Computer Science Department, Stanford University.



- Hayes-Roth, B. (1984)**, "BB1: An Architecture for Blackboard Systems that Control, Explain and Learn about their own behaviour," HPP-84-16, Heuristic Programming Project, Stanford University.
- Hayes-Roth, B. & Hewitt, M. (1985)**, Learning Control Heuristics in BB1," HPP-85-2, Stanford Heuristic Programming Project, Stanford University.
- Hayes-Roth, F., Waterman, D.A. & Lenat, D.B. (1983)**, "Building Expert Systems," Addison Wesley.
- Hayes-Roth, F.A., Erman, L.D., Fouse, S., Lark, J.S. & Davidson, J. (1988)**, "ABE: A Cooperative Operating System and Development Language" in Richer, M. (ed) 'AI Tools and Techniques'. Also in [ref Bond & Gasser].
- Hewitt M. (1977)**, "Viewing Control Structures as Patterns of Passing Messages," in 'Artificial Intelligence', Vol. 8, No.3, pp323-364.
- Hewitt, C. & Liebermann, H. (1984)**, "Design issues in parallel architectures for Artificial Intelligence," in 'Proceedings of the 28th IEEE Computer Society International Conference,' San Francisco, CA. pp418-423.
- Holtman, K. & Kaphan, S. (1995)**, "Problems with the Expires Header," Unpublished web page, <http://www.amazon.com/expires-report.html>.
- Hoare, C.A.R. (1974)**, "Monitors : An Operating System Structuring Concept," in 'Communications of the ACM' 17,10 pp549-557.
- Hoare, C.A.R. (1978)**, "Communicating Sequential Processes," in 'Communications of the ACM,' 21, 8, pp666-677.
- Howe, A.E. (1990)**, "Addressing real-time constraints in the design of autonomous agents," in 'The Journal of Real-Time Systems,' No. 2, pp81-97.
- Huhns, M.N. (1987)**, "Distributed Artificial Intelligence," Morgan Kaufmann.
- Inder, R. (1989)**, "State of the ART: A review of the Automated Reasoning Tool," in Vadera, S. (1989), (ed) 'Expert System Applications,' Sigma Press, Wilmslow.
- INSIGHT (1986)**, "The INSIGHT Blackboard Experiment Information Pack," Systems Designers Ltd.
- Intellicorp (1986)**, "KEE software development system user's manual," Intellicorp Inc.
- Jones, J. (1986)**, "A Blackboard Shell in PROLOG," Report 277, Department of AI, University of Edinburgh.
- Jones, A. (1991)**, The Diagnosis of memory Disorder Related Illnesses using a Multi-Agent Approach," MSc Dissertation, Department of Computation, UMIST.
- Kannan, R. & Dodrill, W.H. (1990)**, "DAIS, A Distributed AI Programming Shell," in 'IEEE Expert,' Decenber.
- Kinney, J.J. (1997)**, "Probability: An introduction with statistical applications," Wiley Press, Chichester, pp198-99.
- Laurent, J-P., Ayel, J., Thorne, F., & Ziebelin, D. (1986)**, "Comparative evaluation of three expert system development tools: KEE, Knowledge Craft, ART" in 'The Knowledge Engineering Review,' December, pp18-29.
- Leao, L.V. & Talukdar, S.N. (1989)**, "COPS: A System for Constructing Multiple Blackboards," in [Bond & Gasser (1988)].
- Lekkas, G. & Liedekerke, M. (1992)**, "Prototyping MAS: A Case Study," in [Avouris (1994)].
- Lenat D.B. (1975)**, "BEINGS: Knowledge as Interacting Experts," in 'Proceedings of the 4th International Joint Conference on AI.'
- Lesser, V.R., Fennell, R.D., Erman, L.D. & Reddy, D.R. (1975)**, "Organisation of the Hearsay II Speech Understanding System," in 'IEEE Transactions on Acoustics, Speech and Signal Processing,' Vol. ASSP-23, No. 1, February, pp11-24.
- Lesser, V.R. & Corkill, D.D. (1981)**, Functionally Accurate/Co-operative Systems," in 'IEEE Transactions on Systems, Man and Cybernetics,' Vol. SMC-11, No. 1 pp81-96.
- Lezak, M.D. (1976)**, "Neuropsychological Assessment," Oxford University Press.
- Liskov, B. & Scheifer, R. (1983)**, "Guardians and Actions: Linguistic support for robust, distributed programs," in 'ACM Transactions on Programming Languages and Systems,' Vol. 5, No. 3, pp381-404.
- Martin, J. (1981)**, "Computer Networks and Distributed Processing," Savant, Carnforth.
- Markatos, E.P. (1996)**, "Main Memory Caching of Web Documents," in 'Computer Networks and ISDN Systems,' Vol. 28, No. 7-11, pp893-905.

- Moon, J.N.J. & Tudhope, D.S. (1995),** "A Multi-Agent Realm for Investigating Navigators' Educational Simulators – Introducing the Marines Testbed," in 'Proceedings of SCSC '95,' July, Ottawa, Ontario, pp949-954.
- Moon, J.N.J. & Tudhope, D.S. (1995),** "MARINES A Multi-Agent Testbed for marine simulation," in 'Marine Technology and Transportation: The Proceedings of MARTRANS '95,' August, Plymouth, UK, pp777-84.
- Muller, J.P (1994),** "A pragmatic approach to modelling autonomous interacting systems," in [Wooldridge & Jennings (1994)], pp226-240.
- Newell, A. (1973),** "Speech Understanding Systems: Final Report of a Study Group," North Holland.
- Newell, A. (1982),** "The Knowledge Level," in 'Artificial Intelligence,' Vol. 18, No. 1, pp87-127.
- Nii, H.P. (1980),** "An Introduction to Knowledge Engineering, Blackboard Model and AGE," HPP-80-29, Heuristic Planning Project, Department of Computer Science and Medicine, Stanford.
- Nii, H.P. (1982),** "Signal-to-Symbol transformation: HASP-SIAP Case Study," in 'Artificial Intelligence,' No. 3, pp23-35.
- O'Hare, G.M.P. (1987),** "New Directions in Distributed Artificial Intelligence," in 'Proceedings of the 2nd International Expert Systems Conference,' London, 30th September - 2nd October, Learned Information Ltd., Oxford.
- O'Hare, G.M.P., Reddy, M. & Jones, A. (1992),** "AMNESIA – Implementing a Distributed Knowledge-Based System using RAPIDO," in 'Proceedings of Expert Systems '92, 12th Annual Conference of the British Computer Society specialist Group on Expert Systems,' (Cambridge, December 1992), Cambridge University Press.
- O'Hare, G.M.P. & Wooldridge, M.J. (1992),** "A Software Engineering perspective on Multi-Agent System Design: Experience in the Development of MADE," in [Avouris (1994)].
- OUP (1991),** "Mini-Dictionary for Nurses," Oxford University Press.
- Operating Systems Review (1981),** 'Proceedings of the Eighth Symposium on Operating Systems Principles, Operating Systems Review,' 15, 5, pp. 64-75, December.
- Parnas, D.L. (1972),** "On the criteria to be used in decomposing systems into modules," in 'Communications of the ACM,' Vol. 5 No. 12, December.
- Pitkow, J.E. & Recker, M.M. (1994),** "A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns," in 'Proceedings of the Second World-Wide Web Conference,' Elsevier, Amsterdam.
- Pollack, M. & Ruguet, M. (1990),** "Introducing TILEWORLD: Experimentally evaluating agent architectures," in 'Proceedings of the 8th Natinal Conference on AI,' Menlo Park, CA.
- Pressman, R.S. (1982),** "Software Engineering: A Practitioner's Approach," McGraw-Hill.
- Price, C.J. (1990),** "Knowledge Engineering Toolkits," Ellis Horewood.
- Rashid, R.F. & Robertson, G.G. (1981),** "Accent: A communication oriented network operating system kernel," in 'Proceedings of the Eighth Symposium on Operating Systems Principles, Operating Systems Review,' 15, 5, pp. 64-75, December.
- Reddy, D.R., Erman, L.D., Fennell, R.D. & Neely, R.B. (1973),** "The HEARSAY Speech Understanding System: An Example of the Recognition Process," in 'Proceedings of the 3rd International Joint Conference on AI,' pp185-193.
- Reddy, M. (1988),** "MICRO-CATCH - A simple blackboard knowledge-based system in PASCAL and PROLOG," MSc Dissertation, December 1988, Supervisor: G.M.P. O'Hare, Computation Department, UMIST, Manchester, UK.
- Reddy, M. (1995),** "RAPIDO: A rapid prototyping toolkit for developing multi-agent systems," CS-95-7, School of Computing, University of Glamorgan, Wales, UK.
- Reddy, M. & O'Hare, G.M.P. (1990),** "COCO-POP A Development Testbed for prototyping Distributed Knowledge-Based Systems," Technical Report AI-90-3 (August 1990), Department of Computation, UMIST University, Manchester.
- Reddy, M. & O'Hare, G.M.P. (1991),** "The blackboard model: a survey of its application" in 'Artificial Intelligence Review,' Vol. 5, No. 3, pp169-186.
- Reddy, M. & Fletcher, G.P. (1997),** "Intelligent Control of Dynamic Caching Strategies for Web Servers and Clients," in 'Proceedings of WebNet '97,' AACE Press,

Charlottesville, 1997, pp.440-446.

**Reddy, M. & Fletcher, G.P. (1998)**, "Intelligent Control of Dynamic Caching Strategies for Web Servers and Clients," in 'IEEE Internet Computing,' IEEE Computer Society Press, pp 78-81.

**Reynolds, D. (1988)**, "MUSE: A Toolkit for Embedded, Real-Time AI," in [Englemore & Morgan (1988)].

**Roda, C. (1990)**, "ARCHON: A Cooperation Framework for Industrial Process Control," in [Deen (1991)].

**Rodden, D. (1988)**, "Cooperation and Communication within an active IPSE," in 'Proceedings of the International Workshop on Knowledge-Based Systems in Software Engineering,' Information Systems Research Group, Department of Computation, UMIST.

**Rosenschein J.S. & Genesereth M.R. (1984)**, "Communication and Co-operation," Report HPP-84-5, Stanford University Heuristic Programming Project.

**Rosenschein, J.S., Ginsburg, M. & Genesereth, M.R. (1986)**, "Cooperation without communication," in 'Proceedings of 1986 Conference of the American Association for Artificial Intelligence,' pp51-57.

**Ross, P. (1985)**, "User Modelling in Comand-Driven Systems," Research Report 264, Department of AI, University of Edinburgh.

**Sellis, T. (1988)**, "Intelligent Caching and Indexing Techniques for Relational Database Systems," in 'Information Systems,' Vol. 13, No. 2, pp175-85.

**Silberschatz, A. (1994)**, Operating System Concepts, Addison Wesley, pp.138-141.

**Sloman, A. (1995)**, "Playing God: A Toolkit for building agents," Research Seminar, Computer Science Department, University of Birmingham, UK.

**Smith, N. (1994)**, "What Can Archives Offer the World Wide Web?" in 'Proceedings of the First International World Wide Web Conference,' Elsevier, Amsterdam.

**Smith R.G. (1978)**, "A Framework for Problem solving in a Distributed Processing Environment," Report HPP-78-28 December 1978, Stanford Heuristic Programming Project, Stanford University.

**Smith R.G. & Davis R. (1981)**, "Frameworks for Co-operative Problem Solving," in 'IEEE Trans. on Systems, Man and Cybernetics,' Vol. SMC-11 No.1 January 1981.

**Sommaruga, L. (1989)**, "An Environment for Experimentation with Interactive Cooperating Knowledge-Based Systems," in 'Proceedings of ther British Computer Society Expert Systems Conference 1989,' BCS.

**Systems Designers (1987)**, "POPLOG user guide," Systems Designers Ltd.

**Waterman, D.A. & Hayes-Roth, F. (1978)**, (eds) "Pattern Directed Inference Systems," Academic Press.

**Werner, E. (1990)**, "What can agents do together: A semantics of cooperative ability," in 'Proceedings of the ninth European Conference on Artificial Intelligence (ECAI-90),' Stockholm, Sweden, pp694-701.

**Wirth, N. (1971)**, "Program Development by step-wise refinement," in 'Communications of the ACM,' Vol. 14, No. 4.

**Wittig, T. (1989)**, "ARCHON - Cooperation of Heterogeneous On-Line Systems," in 'Wissensbasiert Systeme - Proceedings of the 3<sup>rd</sup> International Congress,' Springer Verlag.

**Wittig, T., Roda, C. et al (1990)**, "ARCHON: A Cooperation Framework for Industrial Process Control," in Deen, S.M. (1991) (ed), 'Cooperating Knowledge-Based Systems 1990,' 3-5 October, University of Keele, UK, Springer Verlag.

**Wood, R.A. (1985)**, "Memory Loss: Clinical Algorithms," in 'British Medical Journal,' Vol. 288, pp1143-7.

**Wooldridge, M.J. (1990)**, "The Architecture of Co-operating Intelligent Agents," PhD Transfer Report. Department of Computation, UMIST.

**Wooldridge, M.J. (1990)**, "Towards a formal theory of intelligent social agency: Parts I, II and III" Research Report, Department of Computation, UMIST, Manchester, UK.

**Wooldridge M.J. & Jennings, N. (1994)**, (eds) "Proceedings of the ECAI '94 workshop on Agent Theories, Architectures and Languages," August, Amsterdam, Netherlands.

**Wooldridge M.J. & O'Hare G.M.P. (1991)**, "Deliberate Social Agents," in 'Proceedings of the 10th UK Planning Workshop,' Cambridge, April.

**Wooldridge, M.J., O'Hare, G.M.P. & Elks, R. (1991)**, "FELINE - A Case Study in the Design and Implementation of a Co-operating Expert System," in 'Proceedings of the 11th

International Workshop on Expert Systems and their Applications,' Avignon '91.

**Yang, J.D., Huhns, M.N. & Stephens, L.M. (1985),** "An Architecture for Control and Communication in Distributed Artificial Intelligent Systems," in 'IEEE Transactions on Systems, Man and Cybernetics,' Vol. SMC-15, No. 3, May, pp316-26.

**Young, S.J. (1982),** "Real-Time Languages," Ellis Horewood, Chichester.

### ***Paper 1***

**Reddy, M. & O'Hare, G.M.P. (1991), "The blackboard model: a survey of its application"**  
in 'Artificial Intelligence Review,' Vol. 5, No. 3, pp169-186.  
***(Not included in this binding)***

## ***Paper 2***

**Reddy, M. & O'Hare, G.M.P. (1990), "COCO-POP A Development Testbed for prototyping Distributed Knowledge-Based Systems," Technical Report AI-90-3 (August 1990), Department of Computation, UMIST University, Manchester.**

# COCO-POP\*

## A Development Testbed for prototyping Distributed Knowledge-Based Systems

Mike Reddy<sup>†</sup>      Greg O'Hare

Report AI-90-3  
August 1990

### Abstract

The advantages of a multi-agent approach to knowledge-based systems research are well documented. However, there are problems in developing Distributed Knowledge-Based Systems (DKBSs): which paradigm to adopt, and how the system should be implemented.

Building a prototype application would enable the assessment of the most suitable model for a particular domain. An environment for the speedy construction of multi-agent systems from pre-defined components would facilitate this process.

COCO-POP, a testbed for building prototype multi-agent systems is described which provides a store of low-level primitives, a construction interface and libraries of user configurable components for developing 'off the peg' prototypes. An example blackboard system is then described to demonstrate how an application may be achieved more easily using COCO-POP.

It is proposed that such a tool-based approach to the development of multi-agent systems is vital for the evolution of 'real world' applications. It also serves to emphasise the need for cross-fertilization from the neighbouring fields of concurrent systems and distributed programming.

---

\*Concurrent Control of POP

<sup>†</sup>Supported by a SERC studentship

# 1 Introduction

The material advantages of Distributed Artificial Intelligence (DAI) are well documented and include the scope for modular development and the potential exploitation of multi-processors [Myref]. However the key benefit must be the incremental solution of complex problems by groups of distinct agents (or knowledge sources). A range of paradigms have been proposed [Ref Bond and Gasser], which involve either cooperative or competitive strategies, but most rely upon communications between agents; [Ref Gensereth] is a notable exception.

This reliance upon message passing may require a greater degree of traffic than operating systems or other distributed applications. Different models adopt varying strategies for regulating communications : blackboard systems, for example, use a central database, whereas contract nets negotiate communication channels.

The choice of paradigm for a particular domain application is dependent on many qualitative factors; discussion of which is beyond the scope of this paper. However, there is a clear need for toolkits which support the evaluation of various paradigms without the necessity for low-level programming skills [Ref ABE etc]

The principle concern in building a distributed knowledge-based system (DKBS) is the communications requirement of the adopted paradigm. Although these appear to vary widely, the communications protocols for all distributed systems involve a number of common considerations:

- physical source/destination of messages
- message processing priorities within agents
- synchronization requirements of the system
- pattern of communications between agents [Ref Dist Comp book]

The developer should be shielded from the low-level consequences of these factors where desirable to speed up the prototyping process. Furthermore, it should be possible to construct 'off the peg' applications from pre-defined components supporting a range of paradigms in as simple a manner as possible. A toolkit style interface for applications building would aid this process. Therefore a testbed which supplies these facilities must satisfy the following four requirements:

1. A mechanism for governing a community of agents providing primitives for communications and concurrent execution
2. A high-level command set built upon these primitives to abstract the developer away from low-level considerations unless required.



3. A library of pre-defined DAI constructs for the implementation of existing paradigms or hybrid applications
4. A toolkit for supporting the speedy construction of DKBS prototypes

The rest of this paper will propose a system which attempts to fulfill these requirements. Section 2 will outline the design of COCO-POP and the structure of its constituent parts while Section 3 details their implementation. An application of the blackboard model, the Generic Blackboard Shell (GBS), is then described in Section 4 with emphasis on how this was achieved using COCO-POP's language primitives. Finally, in Section 5, a summary of work to date and an outline of future work are discussed.

## 2 Architecture of COCO-POP

The main objective in the design of COCO-POP is to shield the developer from low-level considerations by an abstraction into high-level constructs. This approach has lead to the identification of a hierarchy of development stages:

1. Constructs for concurrent modules and operators for their manipulation and execution
2. Control structures for a number of different scheduling mechanisms
3. Libraries of high-level constructs based upon these primitives to support various DKBS models
4. Configuration tools to facilitate the construction and configuration of applications

The structure of COCO-POP (shown in fig. 2.0) has therefore been designed to contain four main components :

- A language for representing modules and associated operations
- A scheduler for governing their storage and execution
- A library of pre-defined constructs for use by developers
- A construction toolkit controlled by the user interface.

The design considerations for each of these components is discussed in the following sections.

## 2.1 Modules and Monitors

The key component of COCO-POP is the 'module'; a primitive for the modularization of code. The module has been identified as the primary building block for concurrent systems [Ref Young] as it offers facilities for :

- aggregating data and related operations into logical units
- governing the interaction between separate concurrent processes
- supporting incremental development of systems

Hoare's 'monitor' concept [Refs Hoare, Brinch Hansen], is a special class of module which is a generalisation from the monolithic monitor of operating systems. [Ref Young]. Monitors are used for encapsulating 'critical' operations and devices (such as buffers, shared memory and device handlers) and restricting access to them via mutual exclusion.

Therefore a definition language was necessary to represent the 'module' construct so that separate processes could be created and executed independently. A further extension of this language was required to include a monitor class of modules in COCO-POP. These would be represented in a decentralised manner so that each service or object would have a unique monitor to synchronize operations.

Section 2.1.1 will describe the structure of modules in COCO-POP, while section 2.1.2 describes the operations that may be performed by them.

### 2.1.1 Modules in COCO-POP

A COCO-POP module comprises four main components:

**Module Name** - A unique identifier used for addressing the module during communications

**Module Class** - An identifier used to distinguish between the two types of construct: 'module' and 'monitor'. Modules are expected to execute some function and be freely accessible to others. Monitors are required to serve requests from other modules under mutual exclusion.

**Process Definition** - The principle component of a module which describes local objects and operations available. Furthermore it defines the function that the module is to perform. For a monitor this function is merely the set of commands required for initialization. A module will also contain additional commands defining its main purpose during execution.

**Interface Specification** - This defines the restrictions on inward and outward links between modules and is therefore comprised of two parts: 'public'

and 'employed' lists of operations. These describe respectively those procedures which may be accessed by other modules and those which may be remotely executed within other modules.

A module definition therefore has the logical format similar to that shown in fig. 2.1.1; see appendix A for a full definition.

During execution, modules are expected to undergo different states of execution:

- Running - currently active and executing
- Ready - scheduled for execution
- Sleeping - waiting for some event
- Woken - scheduled to process event
- Sending - passing a message to an agent
- Receiving - processing a message

---

```
<module> := "[" <module_name> <module_class>
           <interface_spec> <process_def> "]"

<module_class> := "module"|"monitor"

<interface_spec> := <employ_list> <public_list>

<employ_list> := "[" { "[" <procedure_name> { <procedure_name> }
                      "FROM" <module_name> "]" } "]"

<public_list> := "[" { <procedure_name> } "]"
```

Figure 2.1.1 - Logical Module Definition

---

### 2.1.2 Module Operations

The functions that may be performed on a module fall into two categories: execution and communication. The execution of modules is controlled with the following primitives:

**halt()** ceases execution and reschedules the current module  
**sleep()** freezes a module until woken or a specific event occurs  
**wake()** unfreezes a module and reschedules it for execution  
**die()** permanently removes a module from activity within the system

Liskov has proposed that communications between modules may be of three types [Ref Liskov from Dist Comp]:

1. A no wait send, where the sending module continues execution once the message has been sent successfully
2. A synchronised send, which blocks the sending module until the message has been processed and acknowledged by the receiving module
3. A remote invocation send which blocks the sending module until the results of the requested operation are returned.

COCO-POP provides for the first and third class of communications with the command `send_to(<message>, <module_name>)`. If the message describes a remote invocation, or Remote Procedure Call (RPC), the operation is performed and the outcome reported to the sending module. Otherwise the message is simply appended to the message queue of the receiving module.

Reception of messages is guaranteed even for 'no wait' sends as the sending module will repeat the message until successfully transmitted. Although there are other methods for handling messages which do not resort to blocking on 'no wait' sends [Ref Rashid and Robertson from Dist Comp], they are less appropriate for DKBSs where correct reception of messages has a higher priority.

Messages are retrieved using the `receive_from(<module_name>)` command. This returns the first message in the message queue received from the named module. If the queue contains no suitable message, execution of the receiving module is blocked until one is received.

Therefore a synchronised receive has been assumed. The provision for a synchronised send and a no wait receive has been made with the signal handler monitor whose implementation is described in section 3. This monitor defines three further communications operators:

`signal(<signal_name>, <message>)` - which blocks execution until the signal is received by another module.

`wait(<signal_list>)` - which suspends execution until a signal from the list has occurred, then receives the associated message.

`is_signal(<signal_list>)` - which evaluates as true if any of the signals in the list have occurred.

## 2.2 Scheduling in COCO-POP

In any model of concurrency there are two possible methods for sharing processor time : synchronous and asynchronous. COCO-POP has adopted the latter approach, where task switching only occurs when a job has been completed or interaction with other modules is required. This is due to the interdependent nature of DKBS tasks, so it would be undesirable to artificially suspend execution at unpredictable stages of task completion. The choice of an asynchronous approach is further supported as it ensures the efficient use of processing time.

Three scheduling strategies have been identified, which complement this model:

1. Round Robin - where tasks are scheduled in order of reception
2. Event-Driven - in which tasks are executed as before but processes reacting to special events are executed in preference to normally scheduled modules
3. Priority-Based - when tasks are executed in a predetermined order of priority

In a priority-based system modules that are scheduled with a higher priority during the execution of another module do not interrupt its processing. However they will be given precedence when the current task is suspended or completed.

Scheduling may only be affected by modules performing the functions described in section 2.1.2. These mark the acceptable points where the scheduler may safely switch between tasks.

### 2.3 Library of agent constructs

These libraries are planned to store routines and constructs for the various DAI paradigms which are accessed by a system configuration tool (next section).

An example of a pre-defined monitor - the signalhandler - (others such as Buffers etc.) built from low level primitives will be shown in Section 3. Section 4 will outline a library of constructs for a blackboard system.

### 2.4 COCO-POP Configuration

A toolkit approach which allows the developer to use 'shell' agents by supplying specific details (such as the knowledge base, scheduling and inference mechanisms, window positioning etc.). At present the system is set to create generic DAI applications, such as a blackboard system. It is envisaged that a more 'expert' user would be able to use constructs indiscriminately to produce hybrid systems.

## 3 Implementation of COCO-POP

To meet the requirements for a concurrent system tailored to developing DKBS applications (outlined in the previous section) certain necessary features have been recognized: facilities for concurrency (even if only low-level); access to languages capable of supporting AI applications; and graphics capabilities for interface design.

Although languages, such as C and Modula-2, offered some of the facilities required, only the POPLOG environment [Ref] provided access to AI languages;

POPLOG programs may be developed in a mixture of LISP, PROLOG and POP-11.

POP-11 provides low-level process switching [ref POP book] which can be used as the foundation of a concurrent environment, while POPLOG offers graphics facilities via the POPLOG Window Manager [Ref] running under Sun-view. COCO-POP was therefore developed in POP-11 running on a Sun series 3 Workstation.

In the following sections the implementation of a monitor for signal handling is described. This is used to demonstrate how COCO-POP primitives may be built up into more abstract functions; detailed in section 2.1.2.

### 3.1 Modules Records

Modules are stored as 'records' (see fig. 3.1a) which have the following components:

**Name** - A unique identifier for the module

**Status** - The current state of execution

**Process** - The process shell for the module

**Window** - An address for the interface window

**Messages** - A queue of messages received by the module :

The process shell of a COCO-POP module is based upon POP-11 process record datatype (similar to those found in Modula [Ref Young] - see fig. 3.1b) supported by a range of pre-defined POP-11 functions. This construct allows POP-11 to switch between separate POP-11 procedures while storing their current status (i.e. values of local variables, the last command executed and the local stack).

In addition each module has an associated interface window created for it. This shows the current state of execution of the module and any other user information that the module may be required to display.

Module records are stored by COCO-POP in module record queues according to their status of execution (see section 2.2). A pointer variable contains the address of the first module in a queue. The rest of the queue is built up as a linked list of module records using three further fields of the module record,

**Priority** - A weighting factor for ordering the queue

**Previous** - The name of the preceeding module

**Next** - The name of the following module

As modules change their state of execution they are stored by the scheduler in one of three queues:

**Runnable** - containing modules that are ready to execute in order of priority

**Waiting** - storing modules that are awaiting some event, such as the reception of a message

**Monitors** - comprising modules that must restrict access via mutual exclusion

A special pointer, **Current**, contains the address of the currently active module. See section 3.2 for further details.

### 3.2 Module Operations

COCO-POP modules are created by the procedure

```
create_module(<name>, <class>, <priority>, <employ_list>,
              <public_list>, <initial_arguments>,
              <procedure_def>, <window_def>);
```

where <priority> is an integer weighting to govern priority-based scheduling; <initial\_arguments> is an optional list of parameters passed to the process of the module; <procedure\_def> is the POP-11 procedure that defines the module process; and <window\_def> is a set of parameters defining an interface window for the module.

The command creates a module, forges links with other modules and initializes its process and places it in the relevant queue for its class; modules would be stored in the 'Runnable' list, monitors in the list of 'Monitors'.

The signal handler would therefore be created with the command

```
create_module("signal_handler", "monitor", 0, [],
              [signal_wait is_signal], [], signal_handler_proc,
              ['Signal Handler' 116 72 960 100]);
```

where `signal_handler_proc` is a POP-11 procedure which defines the enclosed datatypes and operations which may be performed. Once initialized the signal handler module record would be stored in the list of monitors.

When a module sends a message, it does so by switching to the process of the receiving module. The message is then trapped by the `return_remote_call` routine which executes whenever a module process is resumed. This routine determines whether the message is mail or a valid RPC. If the message defines an RPC, the requested operation is performed and results returned to the sending module.

If the receiving module is a monitor that is currently engaged, the sending module will not receive a reply to its call. When the module is next scheduled

for execution it will recognize that no results have been returned and repeat its request to the monitor. This will continue until the monitor is free to respond.

During the processing of an RPC, the receiving module may make nested RPCs to other modules, including the original sending module. However these calls must unravel in the order that they were performed. Otherwise the outcome of a call might be received by the wrong module and other modules would be permanently blocked waiting to receive the hijacked results.

Messages, other than RPCs, are stored in the message queue in order of reception. When the message queue is updated, modules that have been waiting for messages, i.e. sleeping, are rescheduled for execution. With event-driven scheduling, modules receiving messages are rescheduled in preference to other modules so that they can process messages immediately (see section 3.3).

Signals are sent via RPC messages to a server, called the Signal Handler. If modules are waiting for the signal, it is forwarded to them and the signaling module continues to execute. If no module is waiting, the signal is stored within the signal handler and the signaling module suspends itself until it receives an acknowledgement message.

Modules waiting for signals are treated similarly. If a signal has occurred, the waiting module receives the message and the signaling module is released. Otherwise the wait request is stored and the waiting module is suspended until the signal occurs.

### 3.3 Scheduling in COCO-POP

During the execution of modules, their status and position in the system is constantly being altered by the scheduling operations. The status of a module determines where it is stored:

Current - if a module is 'Running'

Runnable - modules that are 'Ready' or 'Woken'

Waiting - modules that are 'Sleeping'

Monitors - monitors that are 'Ready'

Modules or monitors may be 'Sending' or 'Receiving' in any position because chains of nested RPCs may develop at any time.

Modules that are ready to execute are stored in order of priority in the 'Runnable' list so the first module in the list is the next to execute. Therefore the ordering of 'Runnable' is controlled by the scheduling mechanism in operation.



## References for this technical report (in citation order)

- [Myref] Reddy, M. (1988) "MICRO-CATCH - A simple blackboard knowledge-based system in PASCAL and PROLOG," MSc Dissertation, December 1988, Supervisor: G.M.P. O'Hare, Computation Department, UMIST, Manchester, UK.
- [ref Bond and Gasser] Bond, A.H. & Gasser, L. (1988), (eds), "Readings in Distributed Artificial Intelligence," Morgan Kaufmann.
- [ref Genesereth] Genesereth M.R., Ginsberg, M.L. & Rosenschein, J.S. (1984), "Co-operation without Communication," Report HPP-84-36 September 1984, Stanford Heuristic Programming Project, Stanford University.
- [ref ABE etc] Hayes-Roth, F.A., Erman, L.D., Fouse, S., Lark, J.S. & Davidson, J. (1988), "ABE: A Cooperative Operating System and Development Language" in Richer, M. (ed) 'AI Tools and Techniques'. Also in [ref Bond & Gasser].
- [Ref Dist Comp book] 'Proceedings of the Eighth Symposium on Operating Systems Principles, Operating Systems Review,' 15, 5, pp. 64-75, December, 1981.
- [Ref Young] Young, S.J. (1982), "Real-Time Languages," Ellis Horewood, Chichester.
- [Refs Hoare, Brinch Hansen]  
Hoare, C.A.R. (1974), "Monitors : An Operating System Structuring Concept," in 'Communications of the ACM' 17,10 pp549-557.  
Hansen, P. Brinch (1978), "Distributed Processes : A Concurrent Concept," in 'Communications of the ACM', 21,2, pp934-941.
- [Ref Young] *ibid*
- [Ref Liskov from Dist Comp] Liskov, B. & Scheifer, R. (1983), "Guardians and Actions: Linguistic support for robust, distributed programs," in 'ACM Transactions on Programming Languages and Systems,' Vol. 5, No. 3, pp381-404.
- [Ref Rashid and Robertson from Dist Comp] Rashid, R.F. & Robertson, G.G. (1981), "Accent: A communication oriented network operating system kernel," in 'Proceedings of the Eighth Symposium on Operating Systems Principles, Operating Systems Review,' 15, 5, pp. 64-75, December.
- [Ref] Barrett, R., Ramsay, A. & A. Sloman (1986), "POP-11: A practical language for artificial intelligence," Ellis Horewood
- [Ref POP Book] See above [Ref]
- [Ref] *ibid*
- [Ref Young] *ibid*

### ***Paper 3***

**Reddy, M. (1990),** “A Proposal for an ACTOR-Based Extension to COCO-POP – CAST,” Technical Report AI-90-4 (October 1990), Department of Computation, UMIST University, Manchester.

# A Proposal for an ACTOR-Based Extension to COCO-POP - CAST\*

Mike Reddy†

Report AI-90-4  
October 1990

---

\*Coco-pop Actor Shell Testbed

†Supported by a SERC studentship

# 1 Introduction

The development of multi-agent systems has been hampered by the questions of which paradigm to use, and how this model should be implemented. A testbed capable of prototyping multi-agent systems quickly would facilitate the process of evaluation; it has been suggested that this approach is necessary for the viability of future 'real world' applications. Furthermore, this research emphasises the need for cross-fertilization from other areas of computer science; such as concurrent systems and distributed programming.

COCO-POP, a testbed for building prototype multi-agent systems [Myref], has been constructed in an attempt to meet these requirements. Its structure consists of four components:

- A scheduling mechanism for governing process execution
- A core of primitive functions for process execution and interaction
- A library of system components for prototyping DAI applications
- A construction interface to aid application assembly

DAI paradigms may be successfully implemented in COCO-POP as libraries of user configurable components; the first library (GBS<sup>1</sup>) contains elements for building blackboard systems. However, the process involved in developing COCO-POP is iterative, in that changes are driven by the peculiar requirements of each DAI model. Therefore, in order to extend the scope of COCO-POP, further paradigms must be represented for the system to become increasingly generic.

This report describes the results of a literature survey into the ACTOR model of controlling distributed knowledge-based systems (DKBS). In the next section, Hewitt's actor model [Ref Hewitt 1977] is discussed. The construction, requirements and limitations of actors are described. Following this, an outline for implementing actors is proposed. The assumptions and constraints involved in this proposal are then declared. Finally, a rough schedule is given for the construction of an actor library in COCO-POP.

## 2 Actor Theory

### 2.1 The Actor Model

The Actor Model, proposed by Hewitt [Ref Hewitt 1977], was developed to model the parallelism within distributed problem solving. Control of separate processes (or 'actors') was to be achieved by simple message passing, with explicit communication of knowledge between actors.

---

<sup>1</sup>Generic Blackboard Shell

In Hewitt's original proposal, actors were defined as objects combining both procedure and data, hiding their internal representation so that only the external effects (or behaviour) could be viewed by other actors. Actors consisted of two components: a 'script' which defined the behaviour of an actor, and the 'acquaintance list' which defined the neighbours an actor could communicate with.

Computation was expressed as a series of events which defined the causal flow of control between actors. Ordered events, which depended upon some precondition to be calculated, would be executed serially, otherwise events could be treated in parallel. The use of events allowed the concurrency within a problem solving task to be explicitly represented. Completion of tasks was guaranteed by the assumption that they would terminate with 'primitive' actors; i.e. actors that require no communication to satisfy requests.

The script language used for defining actor behaviour allowed for the dynamic creation of primitive actors to perform calculations in parallel; iteration and recursion also require spontaneous creation of actors. It is beyond the scope of this report to give a detailed description of the script language used to define actor behaviour. The following section treats the communications requirements of Actors.

## 2.2 Actor Communications

Communication between actors was via asynchronous, point-to-point message passing. The effect of a message depended upon the scripted behaviour of the receiving actor; similar to speech acts [Ref Wooldridge]. This might cause one of the following effects:

- Calling a remote procedure
- Accessing data
- Modifying data
- Returning a value
- Synchronisation

Messages consisted of a package of data and an envelope which defined the nature of the contents. Envelopes were of three different types: Requests, Replies and Complaints. Requests were produced by actors when they began a dialogue (such as for a procedure call) and request messages would also include an address to return any results to. Replies were messages sent in response to some request. Complaints were messages which occurred if a message was received which did not match the expected pattern, or when its function could not be fulfilled.

Construction of these messages was implicit so the passing of a message could use a syntax similar to regular procedure calls. However it was also possible to state the explicit structure of particular messages. This enabled actors to verify the structure of incoming messages and to redirect them to other actors.

On a single processor, concurrent messages would have to be handled using interrupts. In some cases, there might be a need for strict ordering of tasks; such as for flight booking where reservations and cancellations should be processed without interruption. A 'one-at-a-time' actor has been proposed [Ref], which acts as a guard surrounding the protected actor. This intercepts messages and then forwards them serially to the actor concerned.

The next section describes the proposal that an actor library should be implemented in COCO-POP. Following this some consideration is made of the potential difficulties that may arise during implementation and the assumptions that have been made to avoid them.

### 3 CAST - A Proposal

#### 3.1 Justification for CAST

As mentioned in section 1, further libraries of DAI components are required to drive the development of COCO-POP. GSB, a collection of blackboard system components has already been completed. It is now proposed that a library for representing actors (Coco-pop Actor Shell Testbed (CAST) ) should be developed.

Actors provide a sharp contrast with blackboard systems. Where as the blackboard model is highly centralised, with a globally shared database, the actor paradigm has a fully decentralized control structure. Furthermore, blackboard agents act globally with the blackboard as a focus. Actor systems are autonomous as each may only work in a local context. There is a strong distinction between the methods of cooperation :

Objects [Agents] have to know what to remember.

Blackboards have to know what to forget. [Ref INSIGHT]

Therefore the requirements of implementing CAST will be substantially different to those for GBS, driving the development of COCO-POP itself. In fact, the aims of the actor model (modelling the concurrent nature of problem solving without regard to implementation) closely matches the objectives of COCO-POP! However a number of caveats and assumptions are required for CAST to be developed. These are described in the next section.

#### 3.2 Actors in COCO-POP

There is a potential for actor systems to require the dynamic creation of a large number of primitive actors. This would lead to a large number of very

small grain processes, and consequently an increase in the traffic of messages between them. There is a limit to the number of separate processes that may be controlled by COCO-POP<sup>2</sup>.

However the creation of a primitive actor is both syntactically and semantically equivalent to executing a procedure call [Hewitt 1977] as

...actors which communicate and cooperate with each other in a goal-oriented fashion can be implemented as a single actor. [Ref]

Therefore it would be legitimate to increase the grain size of actors with cohesive collections of procedures representing primitive actions. Although there might be an increase in the redundancy of operations there would be a corresponding payoff with a reduction of coupling between separate processes which would reduce the prohibitive communication and management overheads of a very fine grained system.

Actors themselves are to be represented as modules with initial parameters for scripts and acquaintance lists. Different actor types will be implemented using a range of pop-11 procedures; 'one-at-a-time' actors, for instance, will be represented by COCO-POP monitors, which have the required restrictions upon access.

Primitive actors must be represented by local procedures. Implied message passing by addressing an external actor using a procedure call syntax should be achieved using intelligent communications primitives in conjunction with acquaintance lists.<sup>3</sup>

An interpretation language for reading actor scripts will be necessary to abstract the developer of actor systems from the primitives used for communication. This would be similar to the inference engine found in GBS agents and based upon pattern-matching.

Communications would again be abstracted away from the user by this script interpretation mechanism, with the contents of messages being built automatically unless specified by the developer.

### 3.3 Rough Schedule for Implementation

The following stages have been identified:

1. A script interpretation mechanism and a generic actor definition for the CAST library including procedures for communication and representing primitive actors
2. An extension to the COCO-POP scheduling mechanism for handling actors to improve the concurrent execution of modules

---

<sup>2</sup>14 with interface windows and an upper limit of approximately 900 with respect to storage. Although window space can be recycled, garbage disposal is unable at present to free space from defunct modules so the limit still exists.

<sup>3</sup>This may need some explaining!

3. An extension of COCO-POP to provide a toolkit for producing actor systems from CAST and also to incorporate GBS into one development interface
4. An improved graphical representation of this toolkit and also the execution of modules in general

An tentative deadline for the completion of CAST has been set to the end of January/February<sup>4</sup>. As the first two items are required to demonstrate a working actor system, it is suggested that they are given priority over the others. Having said this, it is understood that the fourth stage could (and should) be at least partially completed in parallel. Therefore the following schedule is suggested<sup>5</sup> :

October & November	: Actor Module Components
November & December	: Improvement of graphical interface
December & January	: CAST and COCO-POP Development toolkit
February	: Optional Overflow time

---

<sup>4</sup>A discussion we had over the summer suggested that this would be a reasonable date.

<sup>5</sup>This is very approximate and needs discussing



### **References for this technical report (in citation order)**

**[Myref]** Reddy, M. & O'Hare, G.M.P. (1990), "COCO-POP A Development Testbed for prototyping Distributed Knowledge-Based Systems," Technical Report AI-90-3 (August 1990), Department of Computation, UMIST University, Manchester.

**[Ref Hewitt 1977]** Hewitt M. (1977), "Viewing Control Structures as Patterns of Passing Messages," in 'Artificial Intelligence' , Vol. 8, No.3, pp323-364.

**[Ref Hewitt 1977]** *ibid*

**[Ref Wooldridge]** Wooldridge, M.J. (1990), "Towards a formal theory of intelligent social agency: Parts I, II and III" Research Report, Department of Computation, UMIST, Manchester, UK.

**[Ref]** Hewitt, C. & Liebermann, H. (1984), "Design issues in parallel architectures for Artificial Intelligence," in 'Proceedings of the 28th IEEE Computer Society International Conference,' San Francisco, CA. pp418-423.

**[Ref INSIGHT]** INSIGHT (1986), "The INSIGHT Blackboard Experiment Information Pack," Systems Designers Ltd.

**[Hewitt 1977]** *ibid*

**[Ref]** See [Hewitt 1977]

### ***Paper 4***

**Reddy, M. (1990),** “CAST – Progress for the Implementation of Actor Communications,”  
Technical Report AI-91-1 (January 1991),  
Department of Computation, UMIST University, Manchester.

# CAST - Progress for the Implementation of Actor Communications

Mike Reddy

Report AI-91-1  
January 1991

Actor communications are often complex and difficult to represent when the standard request-reply format is not applicable. This report discusses the problem of **widow** requests and **orphan** replies within the actor syntax implemented in CAST<sup>1</sup>. A summary of the widows and orphans follows, with an account of how the problem is to be solved using an extension to the existing COCO-POP<sup>2</sup> actor module shell.

---

<sup>1</sup>COCO-POP Actor Shell Testbed - currently under development at UMIST

<sup>2</sup>Concurrent COntrol of POPLOG - currently under development at UMIST

## Current Syntax and Implementation

Reference is made to Report AI-90-4 which explains the structure of CAST. In the current version of CAST<sup>3</sup>, requests and replies may only be represented in matching pairs. Therefore when an actor sends a request, it must then receive back a corresponding reply. Similarly when a request is received by an actor, it must return a reply to the requesting actor.

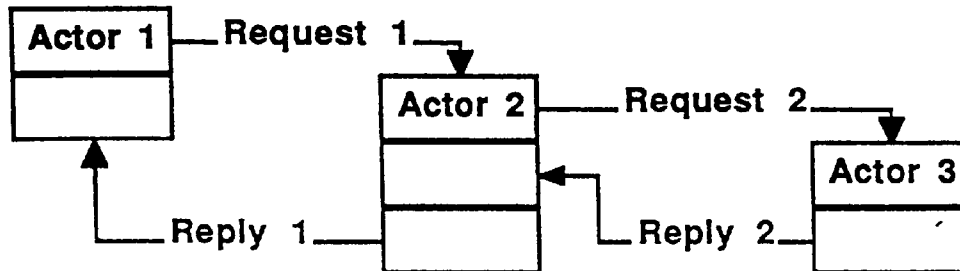


Fig 1 - Request-Reply Pairs in CAST 4.0

Concurrent handling of multiple requests has been implemented using POPLOG processes and a stacking mechanism. Thus an actor responding to two requests will process the most recent request (i.e. last in first out).

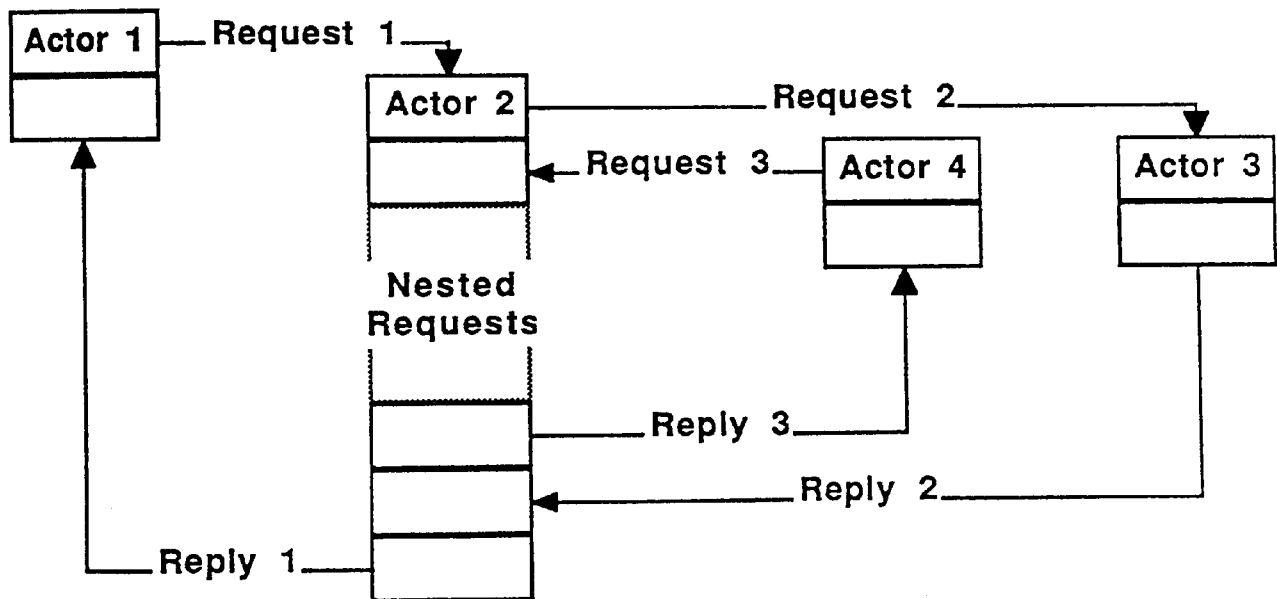


Fig. 2 - Concurrent Request Handling

However, this is insufficient to deal with the other reasons for passing messages, defined in Hewitt's Actor model [Hewitt 1977]: invoking a co-routine, synchronisation and value passing.

---

<sup>3</sup>Version 4.0 running under POPLOG 13.91

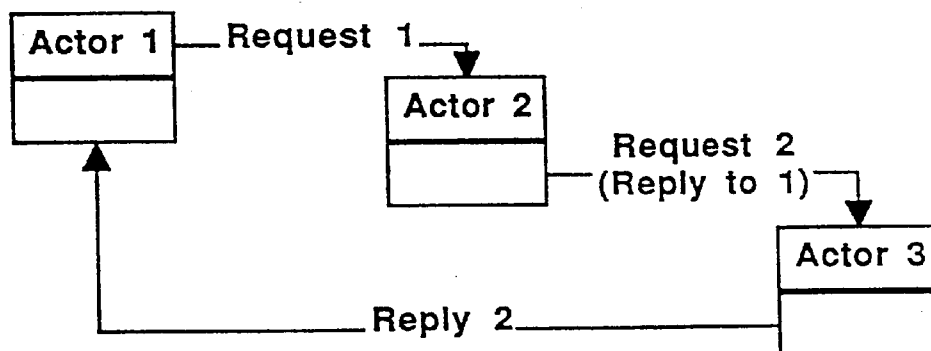
## Widows and Orphans in CAST

The current mechanism forces communications to be synchronous; in that all requests must be answered by subsequent replies. The problem is now described using the concepts of **widow** and **orphan** messages.

Widows occur when an actor sends a request to another actor for which the reply will be redirected to another actor. The receiving actor is determined by the originator of the request using an envelope command so that the reply address can be explicitly stated.

```
[ request [ add 1 2] [ reply_to Actor1 ] ] ~~> Actor 2
```

The diagram below represents two potential problems caused by redirection of requests. Firstly, **Request 2** is a widow because it does not receive a reply from **Actor 3**. Secondly, the final reply does not match with the original request sent by **Actor 1** since it is coming from an unexpected source.

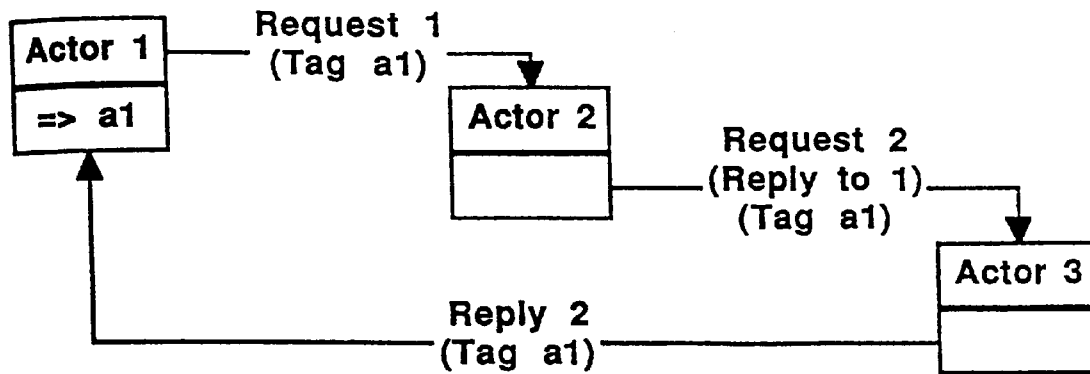


**Fig. 3 - Widow Request**

The current system of stacking, creates a unique tag for each request when it is sent. This tag is also placed on a stack for ensuring that replies are received in the correct order. With the above example, the tag for **Request 1** will not match the tag created for **Reply 2**. Furthermore, this tag would remain on the stack of **Actor 2** indefinitely, as no reply will be received.

In the following example, an orphan, **Reply 2**, is produced by **Actor 3** responding to the widow **Request 2**. For this case, **Actor 4** has no knowledge of this reply and is also expecting a reply from **Actor 5**. Therefore, the difficulty arises in determining the order in which replies should be processed.





**Figure 5 - Passing of an existing tag by a widow**

Primitives for this extended mechanism have been constructed. The next stage is to incorporate them into a script-level system. This entails the building of a script-handler to apply primitives in a formal way, dependent upon a script-like code which defines each actor's behaviour. It is anticipated that this handler system will be implemented in PROLOG<sup>4</sup>.

---

<sup>4</sup>The version of PROLOG envisaged is that supplied as part of the POPLOG environment

## EBNF for CAST Actor Scripts

**actor**:= actor\_name | actor\_definition

**actor\_definition**:=

        "[" actor\_name "<=>" command "]"<sup>1</sup>

**actor\_name**:= identifier

**command**:= send | receive | operation

**send**:= "[" message "->" ( actor | receive ) "]" |

        "[" ( actor | receive ) "<-" message "]"

**message**:= "[" { expression } "]"

**receive**:= "[" "=>" pattern ( command | simple\_expression ) "]"

**pattern**:= "[" { unbound\_variable | expression } "]"

**operation**:= internal\_operation | external\_operation

**internal\_operation**:= "[" prefix\_operator argument "]"

**external\_operation**:= "[" actor argument "]"

**argument**:= { expression }

**prefix\_operator**:= identifier

**expression**:= simple\_expression | operation

**simple\_expression**:= bound variable | constant

**bound\_variable**:= ( "!" | "!!" ) identifier

**unbound\_variable**:= ( "?" | "??" ) identifier

**constant**:= identifier | number

---

<sup>1</sup>Not to be implemented initially.



## **Pseudo Code for CAST Actor Scripts**

### **Create Actor**

Create Acquaintance List

Execute Command

### **Create Acquaintance List**

For each Command in Actor Script do

    If (Command is a Send command) or  
        (Command is an External Operation)

    Then

        Add Actor to Acquaintance List

        If Actor is an Actor Definition Then

            Create Actor

        Endif

    Endif

Endfor

### **Execute Command**

Case

    Command is a Send command

        If Message is a Request then

            Send Request

            Return Result

        Else Message is a Reply

            Send Reply to Actor

        Endif

    Command is a Receive Command

        Perform Receive

    Command is an Operation

        If Operation is External then

            Create Message from Argument

            Send Request

            Return Result

        Else Operation is Internal

            Perform Operation

            Return Result

        Endif

    Command is an Expression

        Evaluate Expression

Endcase

### **Send Request**

Send Request to Actor

Perform Receive

```
If Message is a Reply then
    Return Valid Result
Else Message is a Complaint
    Return Invalid Result
Endif
```

#### **Perform Receive**

Receive Message

```
If Message is a Request then
    Execute Request Script
    If Result is valid then
        Send Reply to Reply Address
    Else Result is invalid
        Send Complaint to Reply Address
    Endif
Else Message is a Reply then
    Execute Reply Script
Endif
```

#### **Evaluate Expression**

```
If Expression is a Simple Expression
;;; Calculate Simple Expression
Else Expression is a Command
    Execute Command
Endif
```

**References for this technical report (in citation order)**

**[Hewitt 1977]** Hewitt M. (1977), "Viewing Control Structures as Patterns of Passing Messages," in 'Artificial Intelligence' , Vol. 8, No.3, pp323-364.

## ***Paper 5***

**Reddy, M. (1990),** “COCO-POP Generic System Capture,” Technical Report AI-91-2 (March 1991), Department of Computation, UMIST University, Manchester.

# COCO-POP Generic System Capture

Mike Reddy

Report AI-91-2  
March 1991

COCO-POP<sup>1</sup> is now at the stage where two paradigms, for blackboards and actors, have been implemented in library form; these are called B<sup>3</sup> (i.e. BBB!<sup>2</sup>) and CAST respectively. Discussion must now turn to the manner in which users may create distributed problem solvers (DPSs) using these constructs..

This discussion document will outline the method by which the structure of an application is taken from the user and transformed into a COCO-POP system.

A model of a possible user dialog is described, with reference to the information needed to define an application at the system and agent levels. Treatment is graphical with some supporting text to explain the procedures.

---

<sup>1</sup>Concurrent Control of POPLOG - currently under development at UMIST

<sup>2</sup>This is a name change because GBS has already been used apparently.

Define the number and names of the agents

User Agent

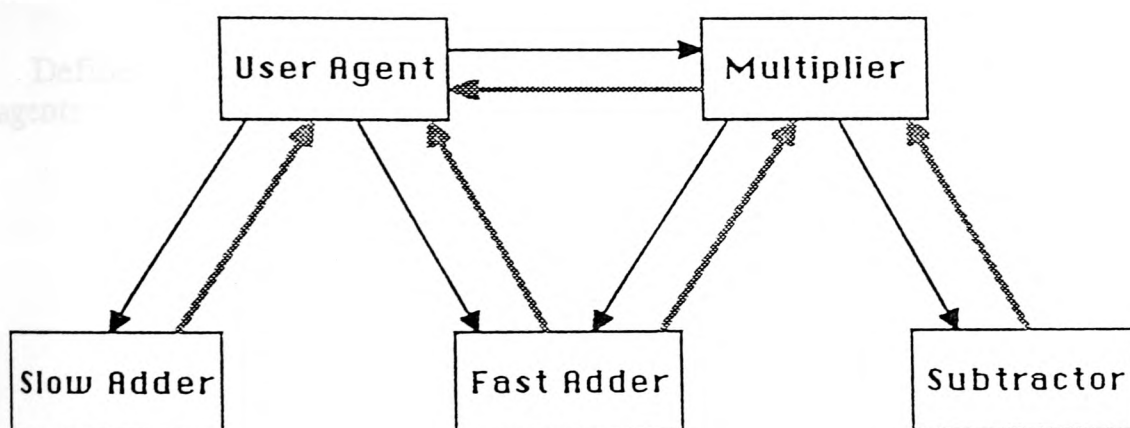
Multiplier

Slow Adder

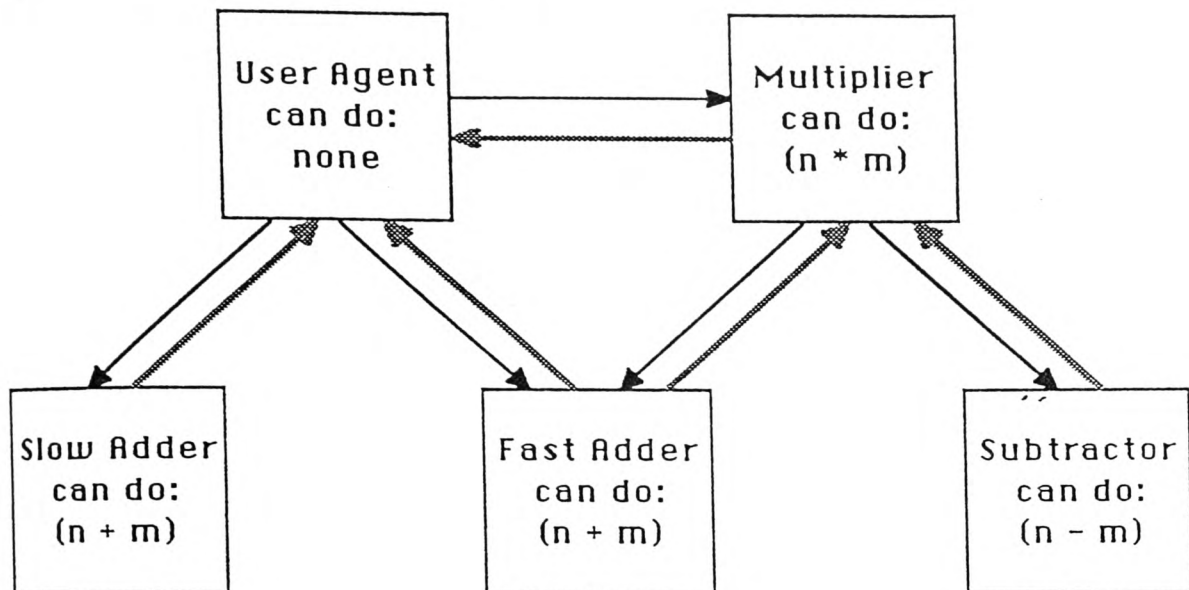
Fast Adder

Subtractor

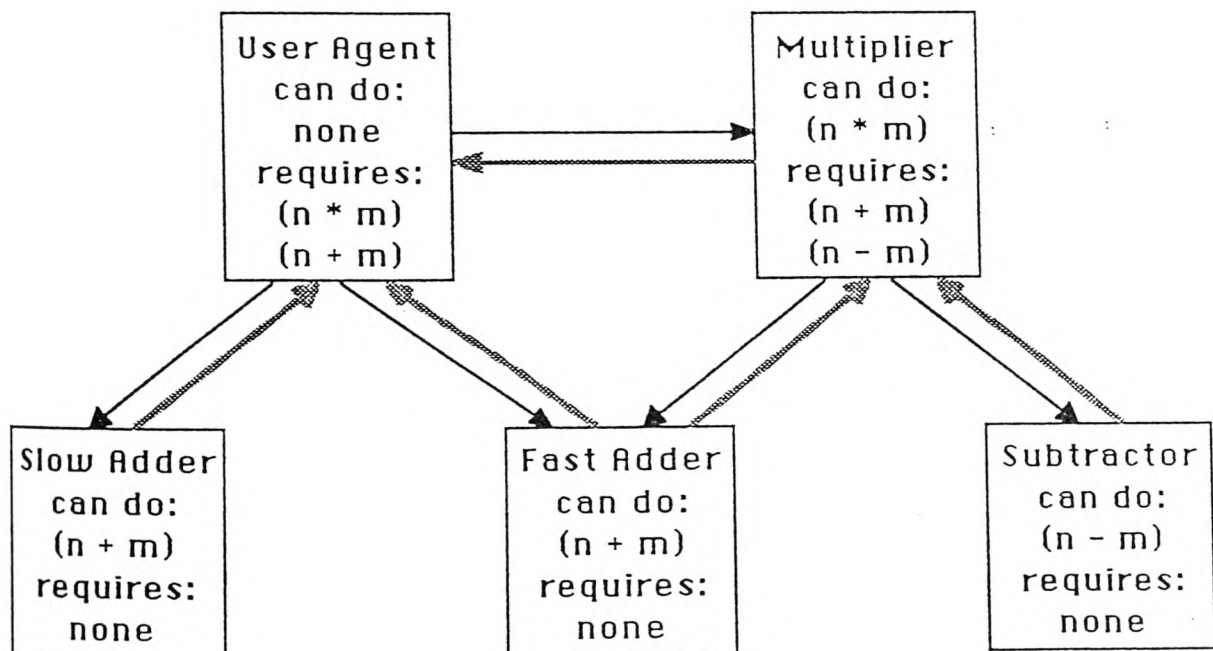
Define acquaintance links between agents - shadow lines indicate that agents may return replies. This is in a 'speak when you are spoken to' arrangement, which allows temporary links between agents when a response is necessary.



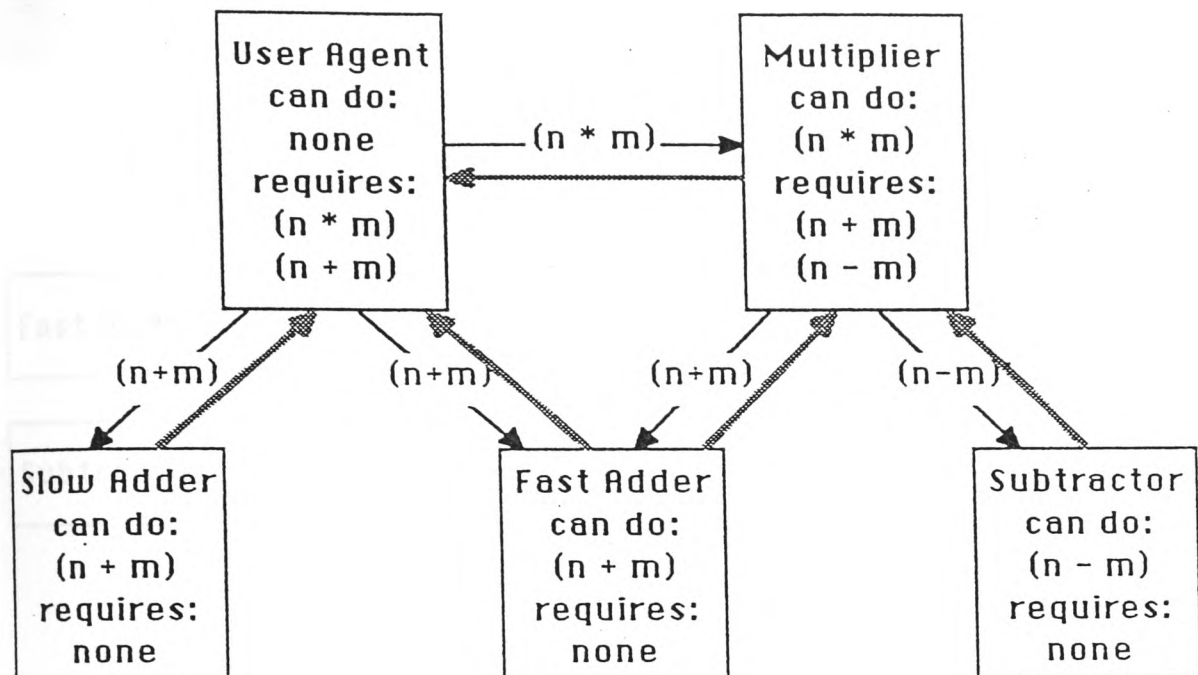
Determine the expertise of agents - i.e. the 'skills'



Define the 'interests' of agents - what sub-tasks will be required by agents

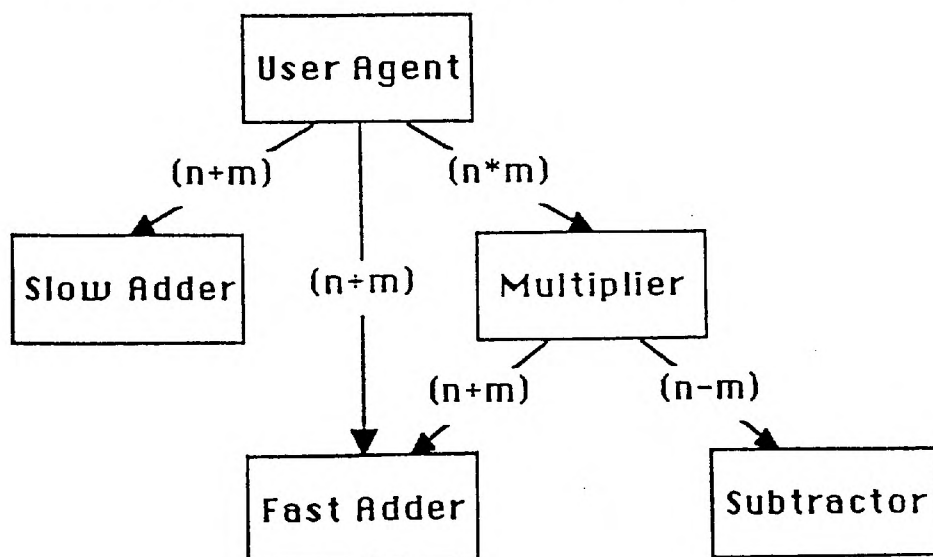


Define potential 'dependency' links between agents using acquaintance and skill information



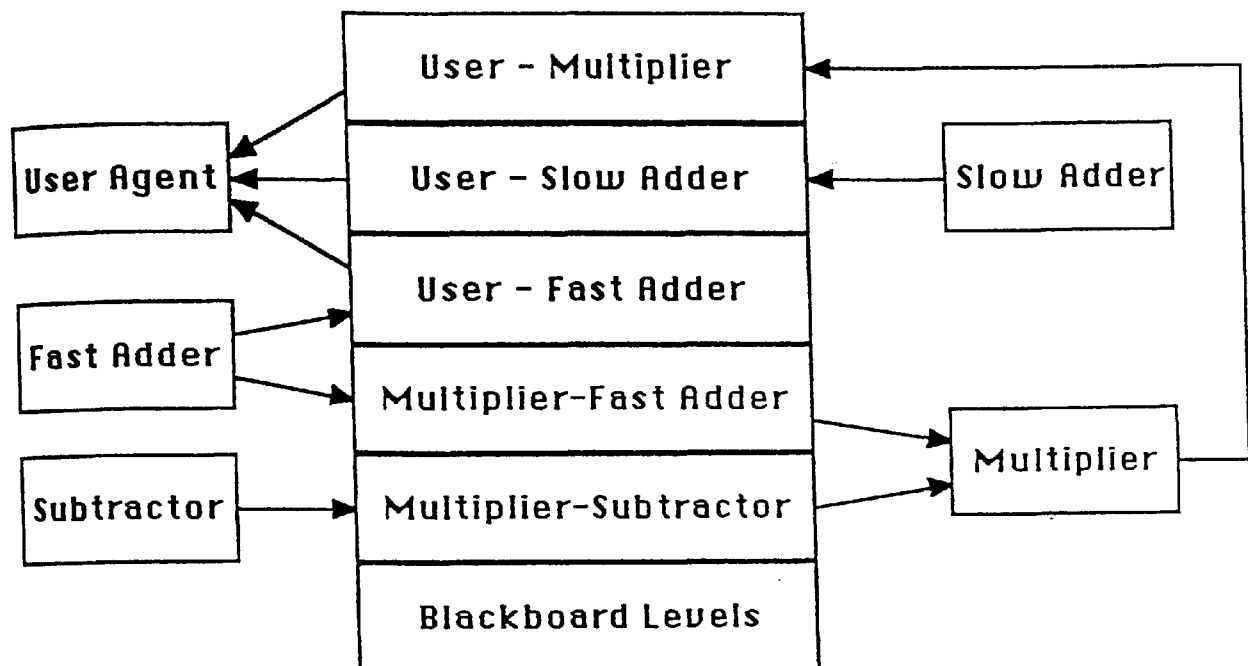
Determine the hierarchy of inter-agent communications, and (potentially) the best paradigm for the system.

Resonably straight-forward for Actor systems





Special treatment of communications via blackboard levels must be examined closely - see supporting material



## ***Paper 6***

**Reddy, M. & O'Hare, G.M.P. (1990), "GARP: A Rapid Prototyping Tool for Distributed Knowledge-Based System," Technical Report AI-92-2 (July 1992),  
Department of Computation, UMIST University, Manchester.**

### **Errata**

P4-5 – Section headed "4 Overview of GARP" should read "3 Overview of GARP" and subsections 4.1, 4.2 and 4.3 should be 3.1, 3.2 and 3.3 respectively.

P6 – Section headed "5 Garp Compilation" should read "4 GARP Compilation" and subsections 5.1, 5.2 and 5.3 should be 4.1, 4.2 and 4.3 respectively.

P7-8 – Section headed "6 Paradigm Specific Libraries" should read "5 Paradigm Specific Libraries" and sub-sections 6.1, 6.1.1, 6.2 and 6.2.1 should be 5.1, 5.1.1, 5.2 and 5.2.1 respectively.

P10 – Section headed "7 Implementation of GARP Applications" should read "6 Implementation of GARP Applications" and sub-sections 7.1, 7.2 should be 6.1 and 6.2 respectively.

P11 – Section headed "8 CoCo-POP Execution of GARP Programs" should read "7 CoCo-POP Execution of GARP Programs" and sub-sections 8.1, 8.2 and 9.3 should be 7.1, 7.2 and 7.3 respectively. Section headed "10 General Conclusions and whinges" should read "9 General Conclusions and whinges"

# GARP: A Rapid Prototyping Tool for Distributed Knowledge-Based Systems

Mike Reddy & G.M.P. O'Hare

Report AI-92-2  
July 1992

This paper will outline a Distributed Artificial Intelligence (DAI) toolkit which enables the rapid prototyping of Distributed Knowledge-Based Systems (DKBSs) for a variety of paradigms – currently blackboard and actor-based systems – This approach is based upon capturing the requirements of the developer by the use of a generic model of agents. This specification is then transformed into paradigm-specific applications for demonstrating first-stage prototypes.

The toolkit is modular in structure, with tools built from low-level primitives in increasing orders of abstraction. Facilities exist for viewing and animating the features of these paradigm-specific prototypes. Further work will aim to increase the robustness of these tools, and provide further facilities for evaluation purposes. Other areas for improvement include the generic handling of interface requirements, and evaluation tools for determining the most appropriate paradigm for particular domains.

## 1 Introduction

It is clear that there has been a trend towards a tool-based approach in the development of software; this trend is most notable within the development of blackboard systems [Reddy & O'Hare, 1991]. Knowledge Engineering (KE) toolkits (such as ART [Ref1], KEE [Ref2] and Knowledge Craft [Ref3]), and similar initiatives in Software Engineering, share the common feature of supporting the developer by shielding the low-level implementation details behind a layer of abstraction. A traditional view of development might involve an iterative combination of three stages:

- 1) Code a prototype with a certain amount of expertise
- 2) Consult an expert to modify or add new knowledge
- 3) Use machine learning to extend the KB 'ad infinitum' [Ref4]

This approach is not without its limitations: Initial prototypes may have to be completely scrapped, after analysis by human experts; reliance upon human experts may cause 'bottle-necks' in the acquisition of knowledge; and machine learning may not be suitable, or even desirable, for the particular domain. The process also requires the presence of a Knowledge Engineer to acquire and interpret 'expertise', and then to perform the programming required to transform this knowledge into an explicit form for the final application. However, the construction and maintenance of large Knowledge-Based Systems (KBSs) is complicated by the increasing amount of explicit data required by more generic applications.

The use of Distributed Artificial Intelligence (DAI) techniques goes some way to alleviating the complexity of development, by distributing knowledge amongst a group of cooperating agents. A number of paradigms have been proposed, which either fall under the 'competing experts' metaphor [Genesereth, 1984 ; Rosenschein & Genesereth, 1984] or the 'benevolent experts' metaphor, where agents cooperate to solve problems.

The latter may communicate by 'task sharing' or 'result sharing' depending upon their communication protocols [Smith & Davis, 1981], while the strategies used to coordinate problem solving have been classified into three distinct groups [Wooldridge & O'Hare, 1991]: Blackboards

(summarised in [Engelmore & Morgan, 1988]), Negotiation. (such as the Contract Net Approach [Smith, 1978]), and Planning [Durfee, 1988].

Although Distributed Knowledge-Based Systems (DKBSs) often require less rules and are easier to maintain, the requirement for even a modicum of programming expertise may dramatically affect the final application:

"The problem [is] that KBSs are often 'programmed' from a low level and do not provide the rich concepts needed for adequate knowledge and control."  
[Berg-Cross, 1989]

An inappropriate representation schema might be forced upon the the developer, due to the influence of the implementation language during development. A further problem arises from the frequent necessity to 'bury' control knowledge, or general 'rules of thumb', by representing control structures implicitly within the programming of an application.

It is clear that some link between the 'application' layer and the 'programming' layer, must be found which shields the user from low-level considerations to prevent implementation detail from dictating the structure of a KBS. However, it must also be suitable for representing low-level algorithms, mid-level heuristics and high-level concepts. This link must therefore exist on the application layer where a user may define the requirements of a system, and its specific domain knowledge without reference to the underlying architecture.

Recent research has therefore been directed towards the development of hybrid systems and testbeds to support the construction of Distributed Knowledge-Based Systems (DKBSs). These range from simple generic shells and DAI architectures (such as ABE [Erman et al, 1988], AGORA [Bisiani, 1987], ARCHON [Wittig, 1989 ; Wittig, 1990], DAIS [Kannan & Dodrill, 1990], RTEK [Feyer, 1990]) to tool-based architectures (including AF [Green, 1987], BB1 [Hayes-Roth, 1984], CooperA [Sommaruga et al, 1989], MACE [Gasser et al, 1987], MADE [Wooldridge, 1990?], MICS [Doran et al, 1991]).

Firstly, this paper will discuss the need for a generic model of DAI paradigms, with reference to their communicational and organisational requirements. In the next section, an essentially pragmatic model of agents is proposed. GARP (Generic Agent Rapid Prototyper), – a toolkit which uses this generic model of agents to aid the development of applications – is then described. Then an explanation is given of how GARP is used to acquire requirement specifications; a detailed description of the structure of these specifications is included. This is followed by an outline of their compilation into paradigm-specific applications, and how the low-level primitives necessary to achieve communication and control are actually implemented for blackboard and actor-based applications. Finally, the merits and failings of GARP are discussed, with reference to future enhancements and improvements, as well as to the lessons learned in its development.

## 2 The Need for a Generic Model of DAI

Multi-agent architectures have a number of common features, independent of the underlying paradigm:

- 1) Applications consist of a group of agents, each having its own 'skills and knowledge'
- 2) The group is assigned 'tasks' to perform, which require decomposition into sub-tasks, to be distributed to the agents.
- 3) Each agent is likely to have access to 'limited' knowledge and resources with which to solve tasks.
- 4) There may be more than one agent which is 'appropriate' for tasks, requiring some form of control. [Cammerata, 1983]

Although all sub-tasks must be assigned to an agent, or agents, for the parent task to be completed, problems may arise when tasks are inter-dependent or where conflict arises over a particular task. Solutions to this 'task coordination' problem must be 'globally coherent' and performed at a local level to be truly distributed. Therefore, agent communications are probably the most important aspect of DKBSs, as they offer the means by which agents cooperate and coordinate themselves.

The scope of communications within existing paradigms, ranges from those which exhibit no communications to those where agents engage in high level dialogues [Werner 1990]:

**No Communication** [Genesereth 1986] - Extensive knowledge of agent beliefs is required, which leads to large overheads in speculation. This approach would only be adequate for non-dynamic applications without the need for sophisticated cooperation.

**Primitive Communication** [Dijkstra 1968, Hoare 1978] - Simple signals or semaphores are used to coordinate and synchronise agents. The limited nature of this vocabulary makes cooperation between agents impossible.

**Plan and Information Passing** [Rosenschein 1986] - Agents mutually decide upon a plan, possibly coordinated by a central planning agent. This approach is computationally expensive and offers no guarantees as no universally acceptable plan may be developed.

**Message Passing** [Hewitt 1977] - Control is represented by the pattern of communications between groups of actors. Sophisticated cooperation is unlikely because the syntax and semantics of these structures are extremely simplistic.

**High-Level Communication** [Cohen & Perrault 1979, Allen & Perrault 1980, Appelt 1985] - Speech acts and agent modeling are used to coordinate agent beliefs and intentions. However there is no formal model of how these techniques may be used to coordinate agent behaviour.

\*Any model of DAI, which claims to be generic, must be capable of representing the diversity of communication protocols available, while modelling the common structures that all paradigms share. It should be noted that approaches, such as agent modelling, dynamically alter their organisational structure during over time. However, even these paradigms should be successfully represented.

Cammerata has proposed that cooperative strategies occur in two classes: 'organisational policies' and 'information-distribution policies' [Cammerata 1983]. Organisational policies govern task decomposition and sub-task assignment. However they also define communication paths for a network of agents, directing and constraining agent behaviour, which might have a significant effect upon the performance of the system. Information distribution policies govern when and how communications occur: subject to imposed constraints from the organisational policy.

Coordination and Control

- 1) Do agents assign their own roles from information received, or have them externally imposed?
- 2) How are roles assigned to agents? – e.g. what strategy is used to determine which agent is consulted about a task.
- 3) Do agents have the ability to negotiate when others try to change their roles or assign new tasks to them?

Communications may vary on a number of dimensions:

- a) broadcast 'v' selective - what criteria are used, if any, to determine which agents communicate.
- b) Unsolicited 'v' solicited - does an agent wait for a request before sending information, or does it control when information is sent.
- c) Acknowledged 'v' unacknowledged - Is reception confirmed
- d) Single transmission 'v' repeated transmission - can messages be repeated and how often (a.k.a. 'murmuring')

Further questions to answer are:

- 1) What does the message contain
- 2) When is the message sent and when received
- 3) How often is the message sent and what is its duration (i.e. effect upon the system)
- 4) What medium is used to deliver the message
- 5) What criteria were used to determine the above decisions

## 4 Overview of GARP

The model is used to provide a framework for highly abstracted elicitation of system requirements.; this is compiled into modules capable of executing concurrently on top of CoCo-POP<sup>1</sup> [Reddy, 1989], an extension of the POPLOG environment [Ref-4]. GARP consists of a WIMP-based intelligent editor to represent, handle and store requirement specifications which represent systems as a set of generic agents<sup>2</sup>, and a compiler to convert specifications into paradigm-specific constructs and produce 'ready to run' code for blackboard and actor prototypes.

### **4.1 Specifying the GARP Model**

#### System level

**Agent editing:** Creation, Naming and Deletion

**System Description**

A blank agent is added to the group, for which a unique name is assigned. It includes a set of undefined stubs for agent and system knowledge.

#### Agent level

**Creation, Naming and Deletion of Skills**

**Defining Visible Skills**

**Defining Acquaintances**

**Agent Description**

Still at a very generic level, the neighbours of an agent which it is allowed to communicate with, must be declared. Acquaintance links are 'one-and-a-half' directional in that acquaintances may only 'speak when they are spoken to'; i.e. a reply may be made without the original agent having to be included in the replying agent's acquaintance list. This method of defining acquaintances does not resolve the means of handling task allocation where there are two or more agents able to satisfy the query. These links are determined at compilation time (see later).

#### Skill level

**Rule creation, modification and deletion.**

**Defining the knowledge and rules that agents have:** including what goals may be proven and the conditions required to prove them. These could be used to determine a likely set of acquaintances for an agent, subject to pruning.

#### Syntax and semantic checking

**Agent level:** At this stage, semantic checks can be performed for name clashes. It should be possible to allow 'stub' agents at this level to reserve places for late additions or modifications. Should an agent's name be changed, or the agent deleted, the other agents which are dependent upon it, are modified. Currently no checking or notification is performed if deleting an agent adversely affects others.

**Acquaintance level:** It is not possible to create an agent at this level, but this facility is to be added. This will allow a more modeless method of agent specification.

**Skill level:** The GARP system does not adequately handle 'orphans'; skills that have no expertise within any of the agents specified.

### **4.2 Structure of Application Specifications**

At this stage, a generic specification of each agent is produced, in the form of a readable text file. This includes descriptive information for documentation purposes, as well as definitions of skills, rules and acquaintance information.

The model consists of a number of distinct agents with: a list of Visible Skills, which are those available to other agents; a list of Acquaintances, which define those agents that may be consulted, and what skills they may offer; and Expertise, includes the knowledge and rules for a portion of the domain.

It has been assumed that agents may only communicate with those others with whom it is acquainted (similar to Hewitt's Actor Model [Hewitt, 1977 ]) or in response to a legitimate

---

<sup>1</sup> Concurrent Control of POPLOG

<sup>2</sup> No discussion will be made in this paper of the best way to split a domain into a series of distinct agents with cohesive knowledge. It's too hard!

The definition of dependency and communication links between agents are then implicitly defined by the set of acquaintances that may be approached to satisfy a query, and the conditions required to satisfy a skill within the agent's own knowledge-base.

Expertise is captured from the developer in the form of production rules. Although this is less abstract than other representation methods, there is a balance between the lower level support for the developer and the power and flexibility of specific knowledge being stored as rules. Production rules are well-known, and are more readable than other more structured representations.

```
rule:= "IF" condition_expression "THEN" action_expression
```

**expression** := constant | bound\_variable | skill\_item | operation | "(" operation ")"

$$\text{skill\_item} := \text{skill\_name} \mid "[\text{skill\_name} \{ (\text{expression} \mid \text{unbound\_variable}) \}]"$$

```
unbound_variable:= "?" variable_name
```

Firstly, skill items consist of either an atom (a simple identifier, with no accompanying argument or attributes) such as 'sea' which may have a value (e.g. 'Atlantic', 'calm', or 32), or a list structure for complex items. An example of this would be [sea north], where north is an identifying attribute, which also has an accompanying value; such as [sea north] := 'stormy'. Although complex items could be seen to be similar to arrays, they have the additional functionality, if required, of deducing some relationship or triggering some action (e.g. [add 5 6] where 'add' is the skill name and '5' and '6' are parameters). This interpretation is controlled by the way skill items are used within rules.

1) Simple boolean rule with no arguments

The skill item 'flu' will therefore have a value 'true' or 'false' (or in a confidence-based implementation, some certainty value), depending upon the result of satisfying the condition section.

IF ~yield < 50 THEN [poor ?yield].

### 3) Rules with complex results or actions

Another example of this would be where the action section did some calculation:

IF (~number\_1 > 0) AND (~number\_2 > 0) THEN [add\_pos\_numbers ?number\_1 ?number\_2] := [add ~number\_1 ~number\_2].

where the calculation would only be performed if the condition is satisfied. If either of the numbers were negative the result would be false. If either were undefined the result would be unproven. In all these cases, skill items used within conditions have not definition of how they are to be deduced, or where. This is defined by the visible skills and acquaintance information of an agent.

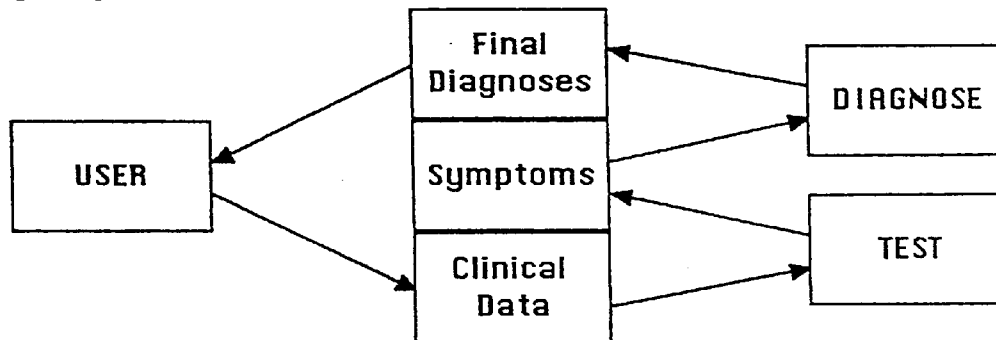
## 5 GARP Compilation

*This section will discuss the consequences of converting a generic model with no consideration of how a problem is to be solved, into a specific paradigm. This section must address the questions raised in 2. The consequences of raising these points will have to be discussed; namely those dealing with the conceptual structure of CAST and BBB, and the implementation of these libraries in terms of CoCo-POP.*

### 5.1 Transforming the GARP Model

Compilation does not produce stand-alone code in one step. Rather an ARM is transformed into a paradigm-specific design specification, which will make use of various libraries of pre-defined constructs; these will be described later.

Therefore, the compilation process is simply a means of transforming generic dependency links and knowledge captured, into a format suitable for a particular paradigm. For example, the previously described set of three agents might be converted into different formats depending upon the final paradigm:



In a blackboard framework, the implicit communication and dependency links might be resolved into a new model, where all message passing occurs via a globally accessible, data structure (the blackboard), with dependency links being represented by unique areas of the blackboard. Access to these areas by knowledge sources would then constitute communications between dependent agents..

### 5.2 GARP Blackboard Compilation

The important components of an ARM used to define a blackboard specification are the:

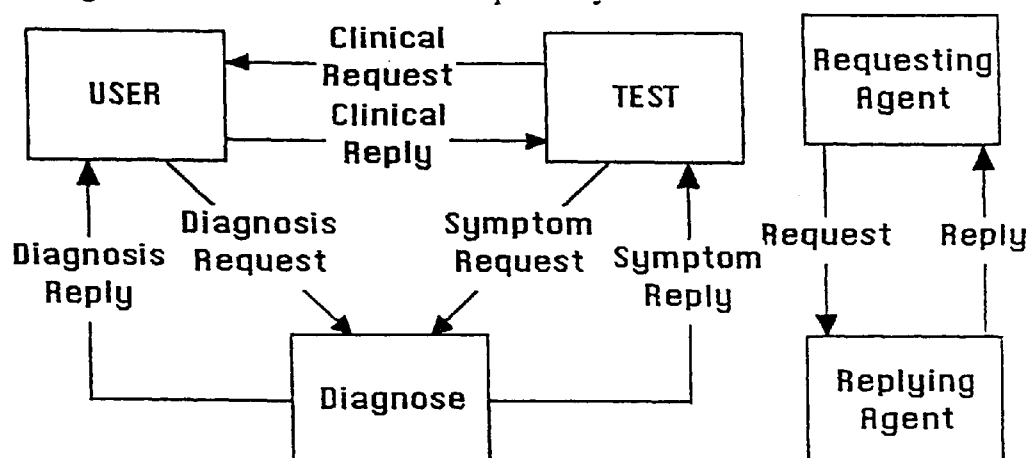
- **Visible Skills** – used in the compilation of the production rules into Condition / Action frames. Conditions of a rule are mapped to unique areas of the blackboard, if visible to other agents, or internally represented if not available. Therefore, the distinction between visible and invisible skills are that the former will be posted on the blackboard, while the latter are internally derived when necessary.
- **Acquaintances** – the dependency of agents for only accessing information relevant to its own skills is determined by which levels it has access to, and which skill items are to be posted there. In conjunction with the visible skills of an agent's acquaintances, unique areas of the blackboard may be created to correspond with the dependency links of the more generic ARM specification.

### 5.3 GARP Actor Transformation

This transformation is more straight-forward to describe than that for the blackboard models, even though the components of an ARM are similarly employed in the transformation into an actor-



based framework. The contrast with blackboard oriented applications is that the actors themselves call their acquaintances for specific information, which is returned immediately (in the same vein as a procedure call would). A further difference with blackboard systems is the need to interrupt the activities of an actor when a new request is received. This is a requirement of actor systems, which is recognised when the case of mutual dependency is considered:



The USER agent must be able to satisfy the query for clinical data made by TEST which was prompted by its request for a diagnosis sent to DIAGNOSE.

Information from and ARM is employed in a similar manner to that of blackboard creation: visible skills and acquaintance lists are used to modify the expertise when it is transformed into actor scripts, so that conditions required of another agent are addressed correctly. Visible skills are represented in scripts as message patterns which when matched with incoming requests will cause the accompanying conditions to be evaluated and the result of the request to be calculated; the syntax of GARP actor scripts will be described in section \*\*\* Therefore, links between actors are 'hard-wired' into the relevant script of an actor at run-time.

NOTE Explain how this is done.

The example for the structure of an actor system is therefore similar to that of the original GARP model. However, an order of events is imposed because the user must make a request, which then triggers further requests, and so on, until the goal is satisfied.

NOTE a real example should be used here.

There is a fundamental difference between the interpretation of this model and that of GARP. Whereas here there is some ordering implied in the sequence of requests there is no temporal information implied in that of the GARP model. This is important because it would be easy to confuse the meaning of a GARP specification, which merely expresses the 'needs and offerings' of agents, with the implicit control path represented by an actor system. Hewitt went to some trouble in his paper to create a formal language for specifying actor systems, and their event-based interdependencies [Hewitt, 1977?]. GARP does not need this temporal constraint as it is a pragmatic and generic method of defining agent relationships without reference to how this is executed.

## 6 Paradigm-Specific Libraries

*This section will provide a discussion of the conceptual structure and organisation of BBB and CAST libraries.*

Although the libraries for actor and blackboard systems (CAST and BBB respectively) contain constructs for radically different paradigms, they are fairly similar in their requirements. Agents are mapped onto modules which include abstracted communication procedures. These operations have been built upon low-level primitives, so that modules may communicate in a number of ways (including mail, remote procedure calls and signals).

The important point to make is that the apparent diversification of agent requirements, when GARP converts its generic model into paradigm-specific applications, is only conceptual. As the upper levels are stripped away, it becomes apparent that both implementations are converging again to a common source

These libraries, and others that may be developed, make use of the same primitives, and yet support a range of contrasting communication protocols. Although their development is in the form of fairly low-level programming, this is invisible to the top-level developer, and even the GARP compiler which merely produces stubs containing the relevant knowledge base for each agent. It is at this level when design becomes implementation, with the first real consideration of low-level construction. As has been shown, GARP succeeds in preventing 'the tail wagging the dog' by effectively shielding users from any programming at all.

## 6.1 BBB (Blackboard Builder)

### 6.1.1 KR for GARP Blackboard Systems

Whereas the transformation of agent communication links into a blackboard framework shows a strong contrast with the original GARP specification, that of the knowledge base is much simpler. The only changes of any significance are: a change from infix to prefix notation, and the substitution of skill items for commands to look up or write entries on the blackboard, or to verify them internally.

Blackboard rules therefore come in the form of:

```
[CONDITION condition_segment ACTION action_segment]
```

Some examples of knowledge represented in this way are:

```
[CONDITION
  [AND [LOOKUP cold ON domain LEVEL] [VERIFY wet]
  ACTION [CREATE flu ON domain LEVEL]]
and
[CONDITION [LOOKUP damp ON domain LEVEL]
ACTION [RETURN wet]]
```

The commands 'LOOKUP', 'VERIFY', and 'CREATE' and 'RETURN' are the first level of an abstract hierarchy of communication which will eventually be reduced to Pop-11 code. Section 4 will describe how these rule operations are implemented by the inference engine which processes blackboard rules within agents. CREATE and LOOKUP are concerned with accessing the blackboard itself, while VERIFY and RETURN are operations which handle internally derived conditions; these correspond to invisible skill items (i.e. ones that are not to be made public by placing on the blackboard).

## 6.2 CAST (Coco-pop Actor System Testbed)

### 6.2.1 KR in GARP Actor Systems

The representation of knowledge in actor scripts is significantly different to that of GARP production rules. This is due to the rich procedural nature defined by Hewitt. Where even the blackboard representation is still fairly abstracted, scripts are far more concrete. Therefore, it can be said that the actor scripts produced by GARP have a greater similarity to the underlying implementation, than the blackboard Condition / Action frames would. the script language adopted by CAST is a subset of Hewitt's own syntax (described in [Hewitt, 1977?]):

```
actor := actor_name | actor_definition
actor_definition3 :=
  "[" actor_name "<=>" command "]"
actor_name := identifier
command := send | receive | conditional | operation

send := simple_send | send_envelope

simple_send := "[" message "->" ( actor | receive ) "]" |
```

<sup>3</sup>Not implemented at present, due to the need for large grained, cohesive agents. These are treated as 'primitive agents' in Hewitt's terminology, so that systems are less dynamic, with less spontaneous creation of actors at run-time; this decision also lends itself to ensuring that conflict does not arise. It could also be argued as a weakness in CAST as there is no mechanism for dynamically updating acquaintance lists as yet

```

message:= "[" ( actor | receive ) "<-" message "]"
send_envelope:= "[" envelope "->" actor "]" |
                "[" actor "<-" envelope "]"
envelope:= "[" "request" message "[" "reply_to" actor "]" |
            "[" "reply" message "]" |
            "[" "complain" message "]"4

receive:= simple_receive | receive_envelope
simple_receive:= "[" "=>" pattern ( command | simple_expression ) "]"
pattern:= "[" { unbound_variable | expression } "]"

receive_envelope:= "[" "=>" envelope_pattern body "]"
envelope_pattern:= "[" "request" pattern "[" "reply_to" unbound_variable "]" |
                  "[" "reply" pattern "]" |
                  "[" "complain" pattern "]"4

operation:= internal_operation | external_operation
internal_operation:= "[" prefix_operator argument "]" |
                   "[" expression infix_operation expression "]"
external_operation:= "[" actor argument "]"
argument:= { expression }
prefix_operator, infix_operator:= identifier

conditional:= rule | case
rule:= "[" "rules" expression { clause } [ "[" "else" body "]" ] "]"
case:= "[" "cases" { "[" clause "]" } [ "[" "else" body "]" ] "]"

clause:= "[" "=>" clause_expression body "]"
clause_expression:= expression | "[" infix_operator expression "]"
expression:= simple_expression | operation
simple_expression:= bound_variable | constant

```

```

bound_variable:= ( "~" | "~~" ) identifier
unbound_variable:= ( "?" | "??" ) identifier
constant:= identifier | number

```

Hewitt's script language [Hewitt, 1977?] is not described here. Rather, the discussion with detail how a production rule is converted into an actor script. The special cases for simple and complex items, with or without parameters and values, which was described in the previous section is also relevant here. When the action section of a rule includes some assignment or calculation, the structure of the accompanying script is different:

```
[=> [flu] [AND [[cold] -> USER] [[wet] -> USER]]
```

This is the case where the result will be a simple boolean value. Note that the return of the result is implicit in the '=>' operation.

```
[=> [add_pos_numbers ?number_1 ?number_2]
    [rules [~true]
            [= [AND [> [~number_1] 0] [> [~number_2] 0]]
            [add [~number_1] [~number_2] 0]]
            [else [~false]]]]]
```

For this example if the condition is satisfied then the addition is performed. Otherwise false is returned.

the communication of agents is performed by the two abstracted operations '=>' and '->' which map onto lower level procedures (described in the next section). These operations cause the

---

<sup>4</sup>Not implemented in this version of GARP

creation of stylised messages which act as requests or replies between agents in the manner of object-oriented systems. The reception of messages, and the response of the receiving actor, are governed by its script.

These messages come in the form of:

```
[REQUEST request_message [REPLY_TO target_name]
  [REQUEST_ID request_id]]
and
[REPLY reply_message [REPLY_ID reply_id]]
```

As can be seen, various slots in actor messages are reserved for system information, such as the originator of a message and a unique identification number for control purposes. These are normally not visible to actors, being for the purpose of organising and synchronising requests.

Actor systems handling of agent communications are context dependent. Actors can send requests and replies and receive requests and replies; the expected format being implicitly defined by the actor script. An example of this is the script samples above where '=>' evokes a 'receive\_request' command, while at the end of the script the value derived is returned to the calling actor with an invisible 'send\_reply' operation. These operations will be discussed in more details in section 5.2

## 7 Implementation of GARP Applications

The prototypes produced by GARP at this stage consist of stub programs which access a paradigm-specific library of modules – BBB is the library for blackboard systems, which supports the creation of large-grain actors which may communicate with each other, via stylised messages.

In short, both these libraries consist of CoCo-POP modules, which have been fitted with inference engines capable of parsing the rule-bases supplied by GARP. The structure of these modules will be described in the next section. Essentially their purpose is to take abstract communications required by either actor or blackboard agents and implement them.

### 7.1 Implementation of BBB

The conceptual structure of a GARP blackboard system consists of knowledge sources which contain expertise in the form of Condition / Action frames, and a global data structure organised into three levels: AGENT, QUERY and DOMAIN. The AGENT level contains information regarding the skills and acquaintances of each knowledge source; this information is used to control the problem solving process. When goals are posted to the blackboard, they are stored on the QUERY level. When this happens the blackboard monitor contacts those acquaintances of the posting agent, which may be able to satisfy the goal. When facts are derived, they are written to the DOMAIN level, where the blackboard monitor will then inform agents which are acquainted with the source of the fact and can make use of the new fact.

This method of partitioning was chosen to prevent the blackboard becoming needlessly complicated. A major feature of the GARP implementation of blackboard systems is the 'intelligent' set of blackboard operations, which implement the necessary information sharing of agents. Therefore agents do not have to actively search the blackboard for useful information: rather they are informed by the blackboard when new information is added, or existing data is modified.

Within knowledge sources, the inference mechanism handles the abstract operations, such as LOOKUP, CREATE, etc, by mapping them onto a series of procedures 'imported' from the blackboard monitor – These are 'read\_entry', 'write\_entry', and 'read\_all'. Additional operations for removing entries from the blackboard are also available: 'delete\_entry' and 'delete\_all'.

The blackboard itself consists of a database of entries which may only be accessed by these procedures from outside. Therefore the blackboard monitor acts as a form of abstract data type. When these routines are executed, the blackboard monitor sends messages to relevant agents in the form of mail.

### 7.2 Implementation of CAST

Unlike the BBB library, all communications for actor systems is, conceptually, a series of structured mail messages. However CAST must implement these as remote procedure calls so that requests may interrupt previous ones, and to allow nested requests to be made (as discussed in

section 2.2). However, this is invisible to developers, as the lowest level seen is that of the actor scripts which maintain the concept of communication via mail.

Rather than communicating with a protected data store, with a view to modifying its contents (as for blackboard systems) CAST actors send messages which are interpreted and acted upon by the receiver. Therefore, each actor has its own set of communication procedures which send or receive structured messages, and perform any necessary pre-processing: 'send\_request', 'send\_reply', 'receive\_request' and 'receive\_reply'. Messages are sent surrounded by envelopes which contain system information, and are normally invisible to the actor script. The above communications command all make use of two lower-level procedures ('send\_envelope' and 'receive\_envelope') which are also available to developers. However GARP-produced actor systems do not make use of envelope commands, as these are not required for the transformation of ARMs into actor scripts.

## 8 CoCo-POP Execution of GARP Programs

### 8.1 Module Creation

The paradigm libraries described above consist of code which supports different conceptual models of DAI systems, providing inference engines for the different representation methods (e.g. actor scripts) and a means of transforming their communication requirements into message passing between modules. CoCo-POP itself consists of module primitives, which allow for the creation and execution of user-defined modules.

### 8.2 Agent Scheduling

These modules are executed concurrently by the CoCo-POP scheduler, while the modules themselves govern safe 'exit' points for volunteering to suspend while others execute.

### 9.3 Communication

The module concept is central to CoCo-POP with each consisting of a set of communication protocols to operate. Each module consists of: pre-specified code (which for GARP is the library constructs defined in BBB and CAST); communications procedures at various levels of abstraction; and an interface definition (which determines which procedures may be imported or exported to other modules). This final component is used to govern remote procedure calls (RPCs), the most primitive of the communication primitives. All other protocols available to developers are built up from this.

Mail messages are performed by the 'send\_to' RPC which places the message in a mail buffer of the receiving module; this also has the side effect of scheduling the receiver for execution, depending upon the priority of the message. This message is then accessed by internal procedures ('message\_waiting' and 'receive\_from').

Further protocols are built up from a combination of RPCs and Mail primitives. For example, CoCo-POP implements signal handling via a system module, called the 'signal\_handler', which allows for synchronised message passing between modules. The procedures 'signal', 'wait', 'is\_signal' and 'is\_waiting' operate on a database of signal messages. This is a similar concept to the implementation of the blackboard monitor in BBB. When using these primitives, a module queries the signal\_handler knowledge base by RPC to see if a signal has occurred, or another module is waiting for its signal. Otherwise the module puts makes a new entry of its requirements, then puts itself to sleep until the signal\_handler informs it of any relevant events via mail.

## 10 General conclusions and whinges

*At present this section merely notes the points to be raised.*

- One conclusion of the paper is that there are few people working on formal theories of distributed problem solving systems. Many problems of DAI are shared with traditional distributed systems.
- Expectation-driven communications is an area of theory at present (Rare!) which would prevent repetition and message redundancy! [Decker 1987]
- Although the approach to modelling DAI systems is simplistic (making no allowance for agent modelling or dynamic mapping of domain knowledge) this pragmatic nature does make itself useful for the speedy definition of systems. It is therefore not a strict formal definition of DAI components, but aims to provide a functional means of specifying domains.

- Although in this case the agents are homogeneous. GARP does not restrict heterogeneous agents from being defined. Nor does the present model prevent agents to be organised into hierarchies in future versions.
- At present, the level of checking for correct complete dependence information is woefully low, but as the system is refined, this will expand to make assumptions about the dynamic links between agents more accessible to the developer.
- As mentioned previously expertise is captured from the user in the form of simple production rules. While this is less abstract than other areas of system specification, the reduction in support for the developer may be overcome in future versions of GARP and do not prove restrictive. This approach does however allow fairly complex knowledge to be represented adequately.
- It should be noted that this restricts the flexibility for dynamically changing agent dependencies, when compared with the blackboard meta-rule system, which is ultimately configurable.
- At present there is no handling for unproven conditions in actor scripts (which are represented in Hewitt's original work by 'COMPLAIN' messages) but this will be added in future improvements.
- Note that this is a feature of Hewitt's original syntax, but it is also possible to make these contexts explicit by using the '==>' and '-->' operations which allowed the script to access the whole message rather than create it in a stylised manner.

## References for this technical report (in citation order)

- [Reddy & O'Hare 1991] Reddy, M. & O'Hare, G.M.P. (1991), "The blackboard model: a survey of its application" in 'Artificial Intelligence Review,' Vol. 5, No. 3, pp169-186.
- [Ref1] Inder, R. (1989), "State of the ART: A review of the Automated Reasoning Tool," in Vadera, S. (1989), (ed) 'Expert System Applications,' Sigma Press, Wilmslow. Also available as AIAI-TR-41.
- [Ref2] Intellicorp (1986), "KEE software development system user's manual," Intellicorp Inc., Mt. View, CA,
- [Ref3] Laurent, J-P., Ayel, J., Thorne, F., & Ziebelin, D. (1986), "Comparative evaluation of three expert system development tools: KEE, Knowledge Craft, ART" in 'The Knowledge Engineering Review,' December, pp18-29.
- [Ref4] Hayes-Roth, F., Waterman, D.A. & Lenat, D.B. (1983), "Building Expert Systems," Addison Wesley.
- [Genesereth 1984] Genesereth M.R., Ginsberg, M.L. & Rosenschein, J.L. (1984), "Co-operation without Communication," Report HPP-84-36 September 1984, Stanford Heuristic Programming Project, Stanford University.
- [Rosenheim & Genesereth 1984] Rosenheim J.S. & Genesereth M.R. (1984), "Communication and Co-operation," Report HPP-84-5, Stanford Heuristic Programming Project, Stanford University.
- [Smith & Davies 1981] Smith R.G. & Davis R. (1981), "Frameworks for Co-operative Problem Solving," in 'IEEE Trans. on Systems, Man and Cybernetics,' Vol. SMC-11 No.1 January 1981.
- [Wooldridge & O'Hare 1991] Wooldridge M.J. & O'Hare G.M.P. (1991), "Deliberate Social Agents," in 'Proceedings of the 10th UK Planning Workshop,' Cambridge, April.
- [Englemore & Morgan 1988] Englemore R.S. & Morgan A.J. (1988), "Blackboard Systems," Addison-Wesley, London.
- [Smith 1978] Smith R.G. (1978), "A Framework for Problem solving in a Distributed Processing Environment," Report HPP-78-28 December 1978, Stanford Heuristic Programming Project, Stanford University.
- [Durfee 1988] Durfee, E.H., Lesser, V.R. & Corkill, D. (1988), "Coherent Cooperation among Communicating Problem Solvers," reprinted in Bond, A.H. & Gasser, L. (1988) (eds) 'Readings in Distributed Artificial Intelligence,' Morgan Kaufmann.
- [Berg-Cross 1989] Berg-Cross, (1989), "Acquiring and managing knowledge using a conceptual structures approach: Introduction and framework" in 'IEEE Trans. Sys, Cyb Man,' Vol. 19, No. 3, May/June, pp513-527
- [Erman et al 1988] Erman, L.D., Lark, J.S. & Hayes-Roth, F. (1988), "ABE: An environment for engineering intelligent systems," in 'IEEE Transactions on Software Engineering,' Special issue on AI. Also in Technical Report TTR-ISE-87-106, Teknowledge Inc., November, 1987.
- [Bisiani 1987] Bisiani, R., Alleva, A., Forin, A., Lerner, R. & Bauer, M. (1987), "The architecture of the AGORA environment," in Huhns, M.N. (ed) 'distributed Artificial Intelligence,' Pitman, pp99-118.
- [Wittig 1989] Wittig, T. (1989), "ARCHON - Cooperation of Heterogeneous On-Line Systems," in 'Wissensbasiert Systeme - Proceedings of the 3<sup>rd</sup> International Congress,' Springer Verlag.
- [Wittig 1990] Wittig, T., Roda, C. et al (1990), "ARCHON: A Cooperation Framework for Industrial Process Control," in Deen, S.M. (1991) (ed), 'Cooperating Knowledge-Based Systems 1990,' 3-5 October, University of Keele, UK, Springer Verlag.
- [Kannan & Dodrill 1990] Kannan, R. & Dodrill, W.H. (1990), "DAIS, A Distributed AI Programming Shell," in 'IEEE Expert,' December.
- [Feyter 1990] Feyter, A.R. (1990), "RTEX: An Industrial Real-Time Expert System Shell," in 'Proceedings of Avignon '90,' Vol. 1, EC2, France.
- [Green 1987] Green, P., (1987), "AF: A framework for real time distributed cooperative problem solving," in Huhns, M.N. (ed) 'distributed Artificial Intelligence,' Pitman, pp153-176.
- [Hayes-Roth 1984] Hewitt, C. & Liebermann, H. (1984), "Design issues in parallel

- architectures for Artificial Intelligence," Proceedings of the 28th IEEE Computer Society International Conference, San Francisco, CA. pp418-423.
- [**Sommaruga et al 1989**] Sommaruga, L. et al (1989), "An Environment for Experimentation with Interactive Cooperating Knowledge-Based Systems," in 'Proceedings of the British Computer Society Expert Systems Conference 1989,' BCS.
- [**Gasser et al 1987**] Gasser, L., Braganza, C. & Herman, N., (1987), "MACE: A flexible testbed for distributed AI research," in Huhns, M.N. (ed) 'distributed Artificial Intelligence,' Pitman, pp119-152.
- [**Wooldridge 1990**] Wooldridge, M.J. (1990), "Towards a formal theory of intelligent social agency: Parts I, II and III" Research Report, Department of Computation, UMIST, Manchester, UK.
- [**Doran et al 1991**] Doran, J. et al (1990), "The MCS multi-agent testbed: developments and experiments," in Deen, S.M. (1991) (ed), 'Cooperating Knowledge-Based Systems 1990,' 3-5 October, University of Keele, UK, Springer Verlag.
- [**Cammerata 1983**] Cammarata, S., McArthur, D. & Steeb, R., (1983), "Strategies of cooperation in distributed problem solving," in 'Proceedings of the 1983 International Joint Conference on Artificial Intelligence,' pp767-770.
- [**Werner 1990**] Werner, E. (1990), "What can agents do together: A semantics of cooperative ability," in 'Proceedings of the ninth European Conference on Artificial Intelligence (ECAI-90),' Stockholm, Sweden, pp694-701.
- [**Genesereth 1986**] Genesereth M.R. & Nilsson N., (1986), "Logical Foundations of Artificial Intelligence," Morgan Kaufmann.
- [**Dijkstra 1968**] Dijkstra, E.W. (1975), "Guarded Commands, Non-Determinism and Formal Derivation of Programs," in 'Communications of the ACM,' 18, 8, Aug 1975, pp453-457.
- [**Hoare 1978**] Hoare, C.A.R. (1978), "Communicating Sequential Processes," in 'Communications of the ACM,' 21, 8, pp666-677.
- [**Rosenschein 1986**] Rosenschein, J.S., Ginsburg, M. & Genesereth, M.R. (1986), "Cooperation without communication," in 'Proceedings of 1986 Conference of the American Association for Artificial Intelligence,' pp51-57.
- [**Hewitt 1977**] Hewitt M. (1977), "Viewing Control Structures as Patterns of Passing Messages," in 'Artificial Intelligence' , Vol. 8, No.3, pp323-364.
- [**Cohen & Perrault 1979**] Cohen, P.R. & Perrault, C.R. (1979), "Elements of a plan-based theory of Speech Acts," in 'Cognitive Science,' Vol. 3, No. 3, pp177-212.
- [**Allen & Perrault 1980**] Allen, J.F. & Perrault, C.R. (1980), "Analyzing intention in utterances," in 'Artificial Intelligence,' Vol. 15, No. 3, pp143-178.
- [**Appelt 1985**] Appelt, D.E. (1985), "Planning English Sentences," Cambridge University Press, New York.
- [**Cammerata 1983**] *ibid.*
- [**Reddy 1989**] First draft of [Reddy & O'Hare (1990)] in Reddy, M. & O'Hare, G.M.P. (1990), "COCO-POP A Development Testbed for prototyping Distributed Knowledge-Based Systems," Technical Report AI-90-3 (August 1990), Department of Computation, UMIST University, Manchester.
- [**Ref4**] Systems Designers (1987), "POPLOG user guide," Systems Designers Ltd.
- [**Hewitt 1977**] *ibid*
- [**Decker 1987**] Decker, K.S. (1987), "Distributed problem-solving techniques: A survey," in 'IEEE Transactions on Systems, Man and Cybernetics,' SMC-17, pp729-740.



***Paper 7***

**O'Hare, G.M.P., Reddy, M. & Jones, A. (1992), "AMNESIA – Implementing a Distributed Knowledge-Based System using RAPIDO," in 'Proceedings of Expert Systems '92, 12<sup>th</sup> Annual Conference of the British Computer Society specialist Group on Expert Systems,' (Cambridge, December 1992), Cambridge University Press.**  
***(Not included in this binding)***

***Paper 8***

**Reddy, M. & Moon, J.N.J. (1995),** "Development and Evaluation of Multi-Agent and Agent-Based Simulation Environments," 'DIMAS '95, Proceedings of the 1<sup>st</sup> International Workshop on Decentralized Intelligent Multi Agent Systems,' (Krakow, November 1995), pp393-401.  
***(Not included in this binding)***

### ***Paper 9***

**Reddy, M. & Fetcher, G.P. (1997),** "Intelligent Control of Dynamic Caching Strategies for Web Servers and Clients," in 'WebNet '97, Proceedings of the 2<sup>nd</sup> World Conference of the WWW, Internet, & Intranet,' (Canada, November 1997).  
***(Not included in this binding)***

***Paper 10***

**Reddy, M. & Fetcher, G.P. (1998), "An Adaptive Mechanism for  
Web Browser Cache Management," IEEE Internet Computing,  
Vol. 2, No. 1, pp78-81, (January-February 1998).  
(*Not included in this binding*)**

### ***Paper 11***

**Reddy, M. & Fetcher, G.P. (1998)**, "Intelligent adaptive web caching using document life histories: A comparison with existing management techniques," in 'Proceedings of the 3<sup>rd</sup> International Workshop on WWW Caching,'  
June 15-17<sup>th</sup>, 1998, Manchester, UK.  
*(Not included in this binding)*

## ***Paper 12***

**Reddy, M. & Fetcher, G.P. (1998)**, "Exp1: a comparison between a simple adaptive caching agent using document life histories and existing techniques," in 'Computer Networks and ISDN Systems,' Vol. 30, Nos. 22-23, pp2149-2153, (November 1998).  
*(Not included in this binding)*